Volume 14
Issue 4
June - August
2010
ISSN 1432-5454

QLToday

The Magazine about QL, QDOS, Sinclair Computers, SMSQ...

Great news:

No price increase for QL Today Volume 15!

Please renew now!

www.QLToday.com

Contents

Please note: the address sheet in the envelope is also the renewal form for the next volume!

The deadline for the next issue is the 10th of August 2010

Advertisers

in alphabetical order

QL Today

ISSN 1432-5454

German office & Publisher:

Jochen Merz Software
Kaiser-Wilhelm-Str. 302
47169 Duisburg
Germany

Tel. +49 203 502011
Fax +49 203 502012
email: smsq@j-m-s.com
email: QLToday@j-m-s.com

Editor:

Geoff Wicks
Flat 5b
Wordsworth Avenue
Derby DE24 9HQ
United Kingdom

Tel. +44 1332 271366
email: gtwicks@btinternet.com
email: QLToday@j-m-s.com

Co-Editor & UK Office:

Bruce Nicholls
38 Derham Gardens
Upminster
Essex RM14 3HA
United Kingdom

Tel +44 20 71930539
Fax +44 870 0568755
email: qltoday@q-v-d.demon.co.uk
email: QLToday@j-m-s.com

QL *Today* is published four times a year, our volume begins on beginning of June. Please contact the German or English office for current subscription rates or visit our homepage www.QLTODAY.com.

We welcome your comments, suggestions and articles. YOU make **QL *Today*** possible. We are constantly changing and adjusting to meet your needs and requirements. Articles for publication should be on a 3.5" disk (DD or HD) or sent via Email. We prefer ASCII, Quill or text87 format. Pictures may be in _SCR format, we can also handle GIF or TIF or JPG. To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hardcopy of all screens to be included. Don't forget to specify where in the text you would like the screen placed.

If you need more information about the UNZIP program which is used by our BOOT program to unpack the files, we suggest that you visit Dilwyn Jones' web site where you find more information about lots of interesting QDOS software and INFOZIP at **http://www.dilwyn.uk6.net/arch/index.html**

Editorial

by Geoff Wicks

I wonder how many readers spotted something missing from the last issue of QL Today.

Usually the penultimate issue of a volume contains a renewal form. It did not appear last time because we needed extra time to consider our future.

Let it be said straight away that we still have our enthusiasm for the magazine. We believe it has a vital role in keeping the QL community together and are eager to continue, but we also have to look at harsh realities. Developments in the QL community could mean troubled times for us ahead.

It should be said that my doubts about continuing were greater than Jochen's. In the past I have always been optimistic about our long term copy position, but now I am more cautious.

Three factors influenced me. Firstly, I had been expecting a decline in the volume of QL news, but it came much more quickly and severely than I had expected. Now there is news to fill only about half the number of pages that we could fill a year ago. Secondly there is the change in the QL-users email group. At one time it was a hot bed of discussion that provided source material about the issues that lived in the QL community. Now it is developing into more of a technical helpline with minimal discussion. Finally there are Quanta's problems. When an organisation of 177 members cannot find itself a treasurer, you have to ask whether it remains viable. If Quanta were to close what effect would it have on the UK scene? What would be the knock-on effect internationally? How would that affect QL Today?

But there is much that remains positive. Both Jochen and I particularly enjoyed producing the last issue. It was gratifying to have so many contributions from outside the UK. It gave us a strong feeling that QL Today is something that must not be lost to the QL community.

This feeling was further reinforced when I did my annual survey of our achievements in the past 12 months. Although a year ago we had promised you about 130 pages, in practice we gave you 172 of which over 86% were editorial. There was a greater variety of topics than I had expected. In total 15 QL-ers wrote for us during the year. Our regular team of writers have remained loyal through two difficult years and for this we are grateful.

At the Quanta AGM I sounded out opinion about the future of QL Today. It was clear that people value the magazine and, more important, wanted it to remain on paper. There was little enthusiasm for an alternative of web publication.

The upshot is that elsewhere in the issue you will find what was missing from the last one - a renewal form. We hope you will make use of it.

But we ask something more. When readers pay for QL Today, they have a right to a quality product. We can only provide that if we get sufficient material for publication as we have no desire to produce either a smaller or a less frequent magazine. Good though our regular writers are, they cannot fill all the pages. We shall still need your occasional contributions.

Ultimately QL Today is your magazine, and we need your help to ensure its future.

QUANTA's Mixed Message

Quanta has 177 members, but not one is prepared to be its Treasurer and once again the organisation faces an uncertain future. Two years ago Quanta's committee feared no one would want to be its Chairman and proposed constitutional amendments with contingency plans for an interim period to continue finding a Chairman before winding up the organisation. Similar contingency plans were not made for the other officer posts of Secretary and Treasurer and at the recent AGM the committee gave no details of how Quanta could survive without a Treasurer.

Three years ago **John Gilpin**, the then Treasurer, wrote in the Quanta Magazine that if no new members were prepared to serve on the committee then Quanta would have to be wound up. Since that warning there has been an increase in the size of the committee. **Dilwyn Jones** joined in 2009 as News Editor and later librarian and last year **Keith Dunbar** as Web Administrator to work alongside Web Master **Dan Abbott**. Waiting in the wings at this year's AGM were two potential new committee members. **Tony Hill** will be the first permanent editor of the Quanta Magazine for almost five years and **David Buckley** will be joining the committee for, as yet, unspecified duties.

Quanta was also able to announce that the recent rapid falls in membership have been stemmed. There was a net loss of just 3 members last year compared with net losses in excess of 11 and 13 in the two preceding years.

At committee level Quanta has become increasingly concentrated in the North of England and Chairwoman, **Sarah Gilpin**, expressed concerns about this in her Annual Report. Last year no committee member lived south of Nottingham and this year only one committee member lives in the south. No less than four of the new committee's seven members are active in Quanta's Manchester subgroup.

Privately Quanta's officers express a mixture of disappointment and anger at the lack of interest in the organisation from members living south of Birmingham. About a quarter of Quanta members live in London and the South East, but only one attended last year's quarter centenary celebration. There was also only one person from the region at this year's AGM. Of the 16 members who attended the AGM 3 came from the South, 5 from the Midlands and 8 from the North. Quanta had deliberately held the AGM at a Midlands venue instead of Manchester to facilitate access for southern members and one officer muttered to QL Today they would have had the same attendance if they had stuck to Manchester.

There has been a further decline of interest in shows. A year ago **Sarah Gilpin** wrote in the Quanta Magazine:

"*The Committee will consider the membership requirement for workshops throughout the country. Perhaps it is time to look at an occasional one day workshop in either the South or the North of the country and expect a low attendance. The Annual General Meeting could then become a regular feature in the Midlands over two days.*"

This year the message was more sober:

"*Unfortunately it is no longer viable to set up two day workshops*".

Quanta's decline in the South East has been rapid. Until the middle of the last decade it was by far the most active Quanta region with regular workshops in Byfleet and Portslade. However the last Byfleet workshop was held in 2006 and the last Portslade workshop in 2007 and there is little prospect of shows being revived in the area. There are still officially 4 Quanta subgroups in the

region, but two of these, Surrey and Sussex, have been without a fixed meeting place for about 4 years.

Quanta's Treasurer crisis comes at a time when the finances of the organisation are likely to come under increasing strain making skilled financial management essential. Technically Quanta has broken even over the last two years but anomalous factors have made it difficult to analyse the true strength of its financial position.

In 2008 there was a financial surplus of £148 but this was because the magazine crisis meant that printing and postage costs were at half of their usual size. In 2009 there was a deficit of £3,050 but this was due to the quarter centenary celebrations for which the committee had allocated up to £3,500 from Quanta's capital. The quarter centenary was celebrated well within this budget and Quanta was also able to finance the special silver jubilee magazine and support for the continental 25 year celebration from the money allocated from the capital.

When the cost of the quarter centenary is removed from the accounts Quanta appears to have had a successful financial year being able to absorb substantial extra costs of producing the magazine, again a knock-on effect from the 2008 magazine crisis, without falling into deficit. Quanta still retains a healthy capital in excess of £9,000.

Quanta's main financial problem is that it has been unable to live from its subscription income for over 7 years. Currently subscription income only covers about three quarters of expenditure. In recent years the gap between subscription income and expenditure has been covered by sales of second hand hardware by **Rich Mellor** on Quanta's behalf, but this is a falling source of income from over £1,000 in 2007 to under £500 last year. QL Today understands Quanta's stock of second-hand hardware is rapidly diminishing and what remains is more suitable for landfill than eBay.

Although Quanta has been able to reduce expenditure on shows and committee expenses, the magazine will continue to be a huge drain on its resources. Printing costs remain relatively stable but postage costs are rising each year and Quanta has been forced to impose an extra postal surcharge on overseas members who do not opt for the electronic version of the magazine. Financially the electronic magazine has not brought the benefits that Quanta had hoped for. Although 2 in 5 of overseas members receive the magazine electronically, only 2 in 14 UK members have opted for the electronic version.

Vintage Computer Festival

Rich Mellor intends to be present at the Vintage Computer Festival to be held in Bletchley Park, Northamptonshire on 19th and 20th June 2010.

He adds:

"This year sees the celebration of retro and vintage computers over a two day Vintage Computer Festival being held at Bletchley Park Northamptonshire, the home of the National Computer Museum and also home of the World War II code-breakers, who cracked the German Enigma machine.

We shall be attending this two day event, with a selection of our stocks - if there is anything in particular which you would like us to bring along, please let us know beforehand."

Full details of the festival are available at:

http://www.tnmoc.org/vcf-gb.aspx

LEAR PCB-DESIGN v7.14

Dilwyn Jones announces a further update of this program:

Formerly known as PCB-CAD, the program has undergone quite a few revisions in a fairly short period of time, including the new name PCB-Design.

The current version is pointer driven, and now that the program's design is stable, the older non-pointered versions have been discontinued.

Here's a list of changes in recent versions:

Version 7.11:

Discrete and SMD libraries updated.

Version 7.12:

Grid colour can be set.

MO-DIP1 library now correctly split into MO-DIP1 and MS-DIP1.

Mouse scroll wheel now changes scale (QPC2).

Build component now demands a valid library ID.

Simplified library component determination. Now just relies on either library ID or component bit set.

Library ID can now only be cleared using 'Krump'.

Version 7.13:

New high density tracking system mode that allows 15 mil track to track spacing with 8 mil tracks and 7 mil gaps.

Gerber and Postscript export supports the new high density mode.
Postscript export now fills all pads.

Version 7.14:
Dot matrix and HP-GL export now support the new high density mode.
Help screen exit works correctly again.
Area layer exchange was global for labels on non library elements. This has now been fixed.
Area handling of library components improved.
New updated logo and name change (PCB-Design) to emphasize continued development.
Logo scales to screen resolution.
Area delete and copy now handle components in library unlock mode correctly.

WEBSITE UPDATES

SELLMYRETRO.COM

Rich Mellor reports:
"The new trading website for retro and vintage computers is starting to gain momentum with over 300 auctions currently listed, we have items listed for the Sinclair computers:
Sinclair ZX81
‹http://www.sellmyretro.com/search?keyword=sinclair+zx81&phraseType=allWords›
and Sinclair QL
‹http://www.sellmyretro.com/search?keyword=sinclair+ql&phraseType=allWords›
home computers.
If you have some items to sell, unlike eBay, we do not charge listing fees, reservation price fees, or Buy it Now fees. We just charge a low final value fee, so you can have a clear out and save greatly on eBay!"

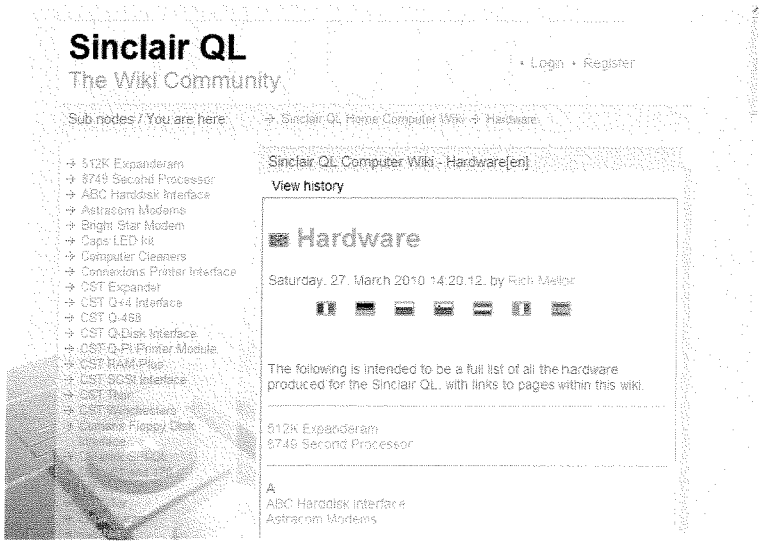
QL WIKI

Rich Mellor has made some major changes to his QL Wiki:
"Thanks to Javier Guerra, I have now uploaded a new theme to give the Sinclair QL Wiki a cleaner and more professional layout. The wiki appears at
http://www.rwapadventures.com/ql_wiki
I have also added some more details on QL software, although this is extremely slow work, as there is only me working on this. Unfortunately, I have more commercial projects in the pipeline on which I need to concentrate my efforts, due to having been made redundant.

It is a shame that Quanta did not feel that it was appropriate to help fund the QL software preservation project, which would have safeguarded as much QL software as possible for current and future QL users - I still feel that this is a very important project, if the QL is to survive for any period, but unfortunately, I need help (financially and time wise) to convert software, test it, rescue software from half-damaged microdrive cartridges and to also get hold of some of the rarer titles which appear from time to time on eBay.
I recently had a look at some of the early Sinclair QL magazines and was surprised by the amount of software and hardware which was listed in the early days, which is now sadly lost (or so it would appear) and still not even mentioned on the QL Wiki"
Later he was able to announce substantial improvements to the hardware section of the Wiki:
"Thanks to Dilwyn Jones, we have now completed a major update to the information on the hardware side to the Sinclair QL Wiki with pictures of most items.
We welcome further contributions to the wiki from the rest of the QL community - maybe this is something that Quanta could usefully promote to its members as a task?
If anyone has any details of further hardware items or spots any errors or omissions, please do get in touch."

QUANTA's New Website

Quanta launched the first stage of its new website at its AGM. At the moment it bears a striking similarity to an expanded version of the old site and there are still more changes to be made.
In his 2009 Website report Quanta's Web administrator, Keith Dunbar, wrote:
"Initially the site will only contain basic informa-



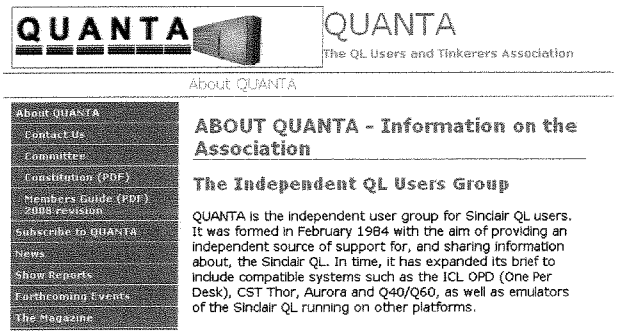
tion, but we do plan to introduce a member's area where the library and the QUANTA magazine will be in accessible form. Also each of the sub groups will have their own page that they can update."

At the moment not all sub groups have responded to Quanta's request for a contact to maintain their own web page.

Keith also adds:

"Once the new site is made live we will also be able to host our own surveys using LimeSurvey."

The current Quanta Survey, reported in the last issue of QL Today, was published under Keith's personal domain for time scale reasons.



One of the problems with Quanta's former websites has been a failure to keep them updated and Quanta hopes that a new content management system will alleviate this problem. In a recent email to traders Quanta's new magazine editor, Tony Hill, reminded them that they should update their web as well as their magazine advertisements.

From the User Group

The QL-users email group provided two lighter items for our news section.

Tobias Fröschle reported that he had discovered a limerick embedded in the code of Computer One monitor:

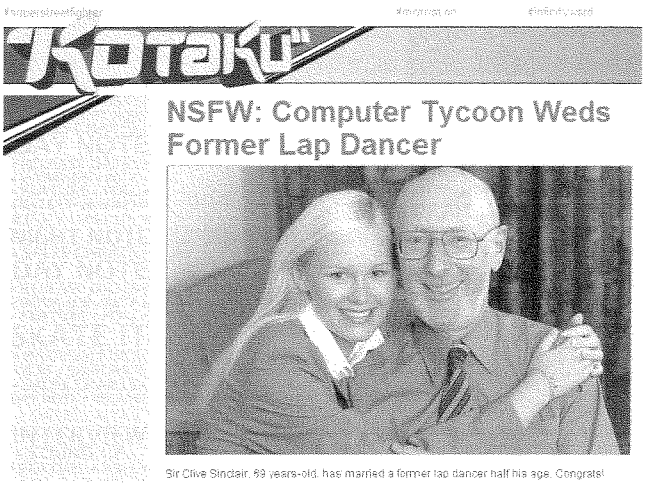
```
dc.b      'A young ma'
dc.b      'n called Paddy Malone',$0a
dc.b      'Decided to try to clone
           clone.',$0a
dc.b      'The mess on the heap',$0a
dc.b      'Caused the Quantum to
           Leap',$0a
dc.b      'And transport him off to
           the Unknown.',$0a,0
```

He wondered whether QL-ers had discovered similar things in other software, but no one replied. QL Today understands that one QL programmer always embeds "QL forever" in his programs even when written for other systems.

Neil Riley came with a juicy piece of gossip:

"Well I never, 69yr old Sir Clive Sinclair has wed a former Lap dancer. She's rather lovely too, lucky Clive!

http://kotaku.com/5525161/nsfw-computer-tycoon- weds-former-lap-dancer



George Gwilt commented "From QL to laptop"

This solved a problem for QL Today's team of forensic experts. In the background to one of the photos QL Today took at the Quanta AGM we noticed something that looked like a crime scene. Was George being mugged? Or arrested? No, just confessing to telling bad jokes!

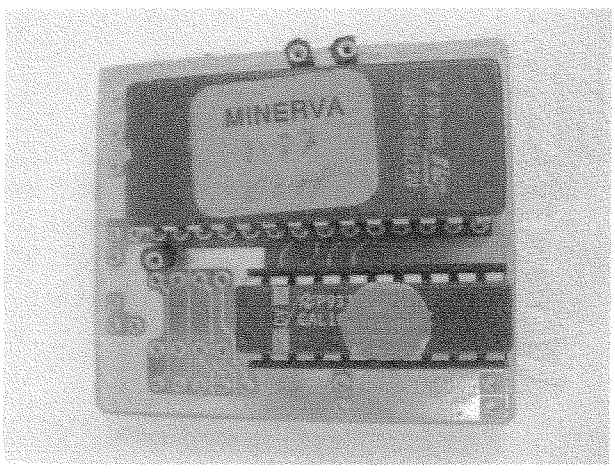
MINERVA Manual Update

Dilwyn Jones reports that thanks to **Tony Firshman**, he has now placed a replacement manual for the Minerva mk I and mk II on his website.

It is available to download in PDF format from

http://www.dilwyn.me.uk/docs/manuals/index.html

(just follow the links at the top down to the TF Services section and find the entry for Minerva).



"QL & MAC are 25" Show

Urs König writes:

"It is a while since the show. But there is still news. The photos (more than 300 pictures) have been online since November 2009 but in good old Sinclair manner it took longer than estimated to complete the aftershow-webpage. ;-) But it's ready now.

With the recent QLvsJAGUAR website update, the QL-Mac-Show page now holds II presentations of the various sessions in both Microsoft PowerPoint (PPT) and Adobe Acrobat (PDF) formats. In addition you can watch related videos on YouTube.

Latest update (May 2010) is that the lost QL software played or hacked with in the exhibition is available on the webpage. This includes Rotating Head Demo by Mark J. Swift, Elite Demo by Dave Barker, QL S*Y*N*T*H by Simon N Goodwin, QTop by Urs König. URL of the QL-Mac-Show webpage is:

http://tinyurl.com/ql-mac-show

Don't forget to have a look at Linus Torvalds (the creator of the Linux kernel) GMOVE software blitter for the QL on the QL-is-25 webpage. URL's are:

http://tinyurl.com/ql-is-25

http://www.qlvsjaguar.homepage.bluewin.ch/QL_Linus_Torvalds.html

In addition to the aftershow stuff I uploaded more pictures and videos of past QL shows to Windows Live Skydrive and YouTube.

Corresponding to QL Today V14 I3, the pictures used in my "The lost treasures in-depth" article are available on my Windows Live Photo Gallery. URL is:

http://cid-c250d8748980ce5a.skydrive.live.com/browse.aspx/Sinclair%20QL/QLmagazines/PicturesUsedInMyArticles

If you're interested in more audio/visual QL resources you may try out one of these URLs:

http://tinyurl.com/ql-story

http://tinyurl.com/ql-pics

http://tinyurl.com/ql-videos

Sad News

Dilwyn Jones writes: **Darren Branagh** got in touch this morning to inform me that his old friend, long time Irish QL User **Hilary o'Kelly**, had a lucky escape and was fortunately only slightly injured when his house in Laragh, County Wicklow, burned down earlier today (13th May) following a fire caused by a defective gas heater

I had the privilege of meeting Hilary several times, both while visiting Darren Branagh when he lived in County Wicklow and on occasions when Hilary and Darren came over to Britain for Quanta workshops.

At the moment, it looks like Hilary may have lost EVERYTHING, including clothing, furniture, computers, etc. His home is so badly damaged by the fire it looks like he may not be able to return there in the short term at least, and is living with Darren's parents for now.

Hilary is 77 years of age and a man of modest means who was unable to afford full insurance, so Darren has asked me if I could make an appeal for equipment to replace Hilary's beloved QL, in the hope that this would perhaps help take his mind off the worst of the situation. If you think you may be able to help in any way, please contact Darren Branagh via his email address

darrenbranagh@gmail.com

TIME TO RENEW

Geoff has already written in his Editorial about the renewal situation. I have to write about the financial side, and I have written about the considerations more or less every year. Printing costs could go up, postage is quite likely to go up at the beginning of next year, as it does in Holland more or less every year etc.

We have given you a lot more pages than we promised at the beginning of this volume (including the current issue), but the calculation was based on having to send two issues from Austria and two issues from Germany. It worked out that I was able to send all issues from Austria, where the postage is lower ... and this made up for the extra pages.

As far as I can work out right now, I should be able to send at least two issues of the next volume from Austria. Assuming an average of 32 pages (or a few more, with your help) per issue, and praying that Austria does not raise the postage price, I don't need to care about Holland's postage increase as I won't send anything from Holland anyway - or at least only two issues. It is a calculation with so many factors out of my control - like it was in the past - that it is not a real calculation anyway.

It is more a "guess and hope", and as you can imagine, there's not been much to earn for some time now. All the base costs (like printing the masters, driving to the printer, picking up the magazine when it is ready) are the same, or actually higher every year, and even more unfortunately, for fewer readers. I am glad to still have so many trusty readers, but there are factors which nobody can control which lead to a slightly shrinking readership ... after all, the QL has existed now for 25/26 years and we are all getting older (for those who have not met me for a while, I really turn grey!).

So, good news: I decided to leave the price for the subscription as it was last volume, in EUR. Even better news for our readers in the UK and USA: the exchange rate is much better now, "thanks" to the recent very bad news about the EUR. Do you remember my criticism about the EUR when it was introduced? I did not even think at that time that it could turn into something so big...

As I don't know how the currency is going to go, I have to add some kind of "buffer" to the exchange rate ... but it will still be cheaper for non-EUR readers than it was last year.

Now a final plea from my side: I did everything I could do to keep the costs low, and there is one thing you can do: don't wait for reminders, renew now. There's a form at the end of the magazine, which you can copy, and there's also a separate sheet with the address label to make it even easier for you.

Sending out reminders is very expensive - a letter to the USA costs EUR 1.70 in postage - which is way over 2 US$ (at least still at the time I write this).

So please help in keeping the costs of the magazine down ... reminders are really expensive, therefore...

PLEASE RENEW NOW!

BASIC Programming Aids - Part 2

by Dilwyn Jones

CD and EDIT

This is a pair of extensions to BASIC written by Malcolm Lear. It actually comes as two separate extension files. The first includes CD and CDp for changing data and program defaults. The other, the one I am interested in for the purposes of this article, is a redefined EDIT command.

The new EDIT command can behave like the old one, but also allows you to specify a string in place of a line number. The string is the name of a procedure or function. EDIT then starts editing at the line at which that procedure or function is defined. Nice and simple - the edit command retains its previous facilities but now also gains a useful new facility.

To install the extension, just LRESPR the file called EDIT_BIN and that's it! Those into assembler can even study the source file, EDIT_ASM.

A nice, neat and simple little aid for the S*BASIC programmer.

It can be downloaded from the Toolkits page on my website at

http://www.dilwyn.me.uk/tk/index.html

DIY Toolkit

Simon Goodwin has written a number of very useful little extensions which can be of great help to the S*BASIC programmer. As you would expect from Simon, they are well written, well documented, very simple to use and you get the source files as well if you want to have a look at how it was written.

The DIY Toolkit series appeared in QL World magazine over a number of years and Simon gathered them into a series of 24 volumes on disk, identified by a letter of the alphabet. When you get the DIY Toolkit, all you have to do is identify the volume for a particular extension or set of extensions and add them to your collection of toolkits and extensions and add the appropriate RESPR/LBYTES/CALL or LRESPR commands to your boot programs to install them.

You can download the entire DIY Toolkit series as a set of three zipped disks from my website at

http://www.dilwyn.me.uk/tk/index.html

or from Simon's Qdos DIY Toolkit page at

http://simon.mooli.org.uk/QL/DIY_Toolkit.html

Simon's page includes a brief list of what is in each volume, and I'm only going to mention a few which I have found useful, without going into full detail of every single extension, which would probably fill up a few volumes of the magazine!

Volume A - ALIAS and DEF$

ALIAS is an extension which lets you change the names of existing QL commands, a simple reason for this might be someone whose first language is not English wishing to rename QL commands to his or her own first language, e.g. ALIAS "OPEN" TO OUVRE for a French speaker. These name changes can survive a NEW or LOAD and remain available throughout a session, but would obviously need to be redefined again after a reset or switch-on. ALIAS can only change the names of resident procedures and functions (not keywords in ROM like PRINT or END or REPeat), but this in itself can be very useful. Its use is restricted to environments which feature a name table, so interpreted SuperBASIC is OK, but Supercharged or Turbo compiled programs would not allow the use of ALIAS inside the compiled program.

ALIAS forms a set when used with some other DIY Toolkit keywords such as FORGET in volume B, but is quite useful by itself.

_DEF$ and _DEF% are extension functions to help with editing procedures and functions by name rather than line number. Jan Jones and Tony Tebby went to great lengths to give us a version of BASIC which is highly structured and not dependent on line numbers if you avoid keywords like GOTO and GOSUB, so it is handy to have extensions like this which help with editing programs in a structured way.

The principle behind these functions is that they bring up a list of procedure and function names in the BASIC program in memory and you choose the one you want to edit from a menu presented in the screen window channel you specify. Having done this, _DEF% returns the line number of the definition which you can use in an ED or EDIT command without having to look at listings and printouts to find the line numbers you want to edit. A huge advantage with long S*BASIC programs.

It comes into its own when you follow Simon's suggestion in the documentation and define a convenient ALTKEY or perhaps a HOT_CMD if using the Hotkey system which simply issues a quick ED _DEF% command for rapid editing via a menu. Something like ALTKEY 'e','ED _DEF%' & CHR$(10). By default, the menu is presented in screen window #2, like the ED command, but you can specify a different channel number if you wish, e.g. ED _DEF%(#1)

Figure 9 shows a fairly typical ED _DEF% menu, where the names are listed in #2. In fact, there can be more names than the number of lines in the window, the extension will simply scroll the menu up and down as required. Just use the cursor up and down arrow keys to move the highlight to the name of the routine to be edited and press ENTER. If names are too wide for the window, they just get truncated in the list.

The _DEF$ variant just returns the name of the routine selected - this might come in useful with Malcolm Lear's extended EDIT command described earlier.

Installing this very useful extension is simply a matter of LRESPRing the file called DEFS_CODE. I tend to have this included in my main boot program whatever I'm doing, I find it so useful, especially for long programs with so many routines defined I forget some of the names, and it also saves a lot of typing - which would you find easier in the example shown in Figure 9,

```
ED QFIND('Select_System_Palette')
```

or just

```
ED _DEF%?
```

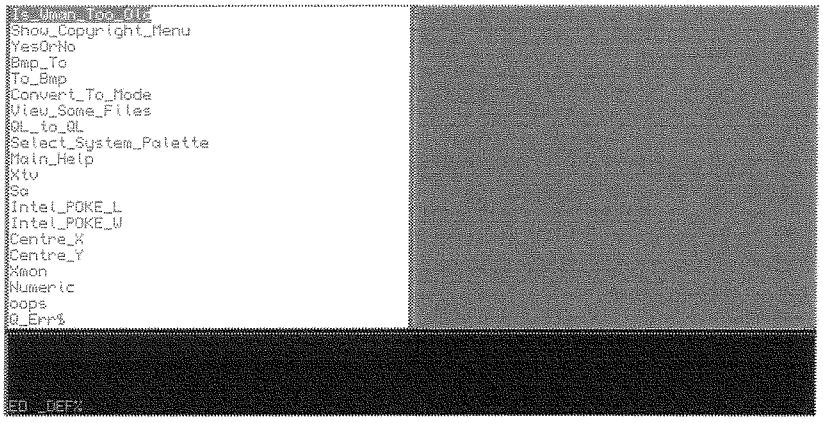
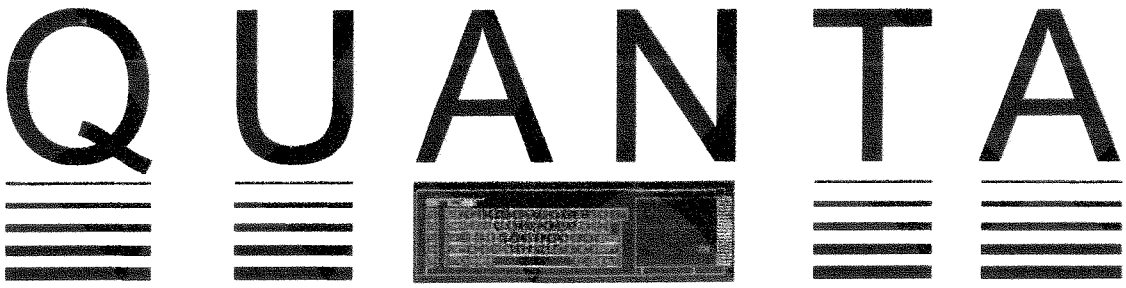


Figure 9 - A typical _DEF% edit selection list

Volume M - MultiBasic

I'm not going to say too much about this one, as it's not a programming aid per se, but if you are a QDOS user and envious of the multiple SBASIC jobs you can create in SMSQ/E or the multiple BASICs in Minerva, this DIY Toolkit set will give QDOS or Thor Argos users multiple BASICs as well. In other words, have multiple BASIC programs in memory at the same time. Very handy!



Independent QL Users Group

World-wide Membership is by subscription only,
Offering the following benefits:
Bimonthly Magazine - up to 52 pages
Massive Software Library - All Free!
Free Helpline and Workshops
Regional Sub-Groups. One near you?
Advice on Software and Hardware problems
Subscription just £14 for Full Membership

PayPal (see QUANTA Web Site),
Cash, Cheques and Postal Orders Accepted

Now in our Twenty Sixth Year

Further details from the Membership Secretary

John Gilpin, 181, Urmston Lane,
Stretford, Manchester, M32 9EH (UK).
Tel. +44 (0) 161 865 2872
Email: membership@quanta.org.uk

Visit the QUANTA Web Site
http://www.quanta.org.uk

Volume R - Replace

Ever been in the situation that you regret using a particular variable name, perhaps because it clashes with the name of an extension in that new toolkit you start using? Yes, you have to go through your program changing the variable name by hand which can take ages and is error-prone. You can't be certain you've found and replaced all of the variable name references.

Which is where the REPLACE extension comes in. LRESPR the extension, load the BASIC program and it works on that program in memory. Suppose you've used the variable f$ and want to change it to a more meaningful name such as filename$.

```
REPLACE f$,filename$
```

Replace only works with unquoted variable names, including array names. Try to change procedure or function names with this extension and you may get an error due to the way that Super BASIC parameter passing works. But despite this limitation it's still a great facility for changing variable names automatically throughout a BASIC program.

Note that if your program uses Minerva Integer Tokenisation, you may need a second version of REPLACE, which is supplied on the disk to cope with the slight changes introduced by the Minerva ROM in this respect.

Volume X - VOCAB

If you've ever been frustrated by the limitations of the EXTRAS command for listing extension names installed on your QL, consider using Simon's VOCAB extension instead. It can list names in columns in a given screen window and can list the 10 SuperBASIC name types (from 0 for unset names such as device names or filenames, to 9 for resident functions) separately if required. If you don't specify a name type, all are listed. The syntax of the command is

```
VOCAB [#channel,] [name_type_number]
```

The channel number and type number are both optional - default is channel #1 and all name types.

DIY Toolkit should be mandatory for all BASIC programmers. Even if you don't use many of the extensions provided, you can still learn quite a lot by reading Simon's documentation and code.

QCOPY

Barry Ansell wrote this little routine hijacker called QCOPY, which lets you copy routines out of a QSAVEd file. Normal BASIC programs are saved as plain text files, which can be viewed just by copying them to the screen, for example. QSAVE was originally a utility by Liberation Software, a companion extension to the QREF utility mentioned last issue. These days, SBASIC on SMSQ/E systems has a compatible extension of the same name too. QSAVE was originally written to speed up the saving of long BASIC programs, or rather to save them in a format which could be reloaded with QLOAD much more quickly on QDOS systems than the rather slow loading process of LOAD, which had to retokenise the program line by line as it loaded. However, BASIC programs saved with QSAVE were tokenised and not in viewable ascii form, so it made it rather difficult to look at and extract routines for inclusion in other programs.

So QCOPY loads a QSAVEd BASIC program, which will normally have the filename extension _sav, and lets you play with that file. It can list information about the content of a _sav format BASIC program - see Figure 10 for an example. You can choose which proc or fn to extract, and choose to send output to screen, printer, file or the SCRAP system.

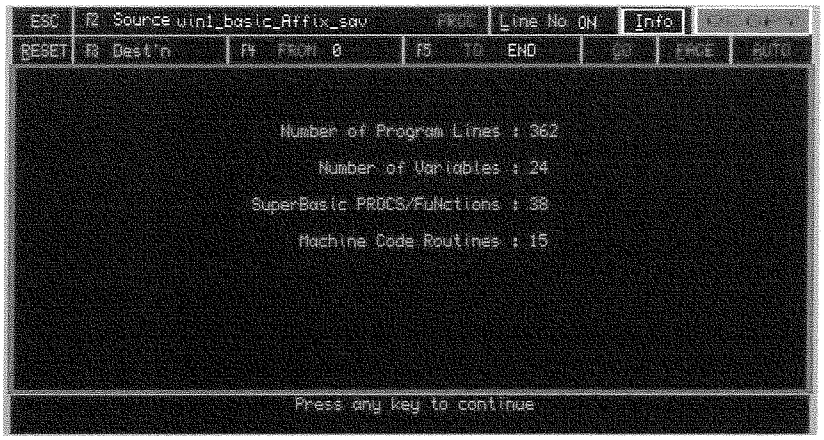


Figure 10 - Example of QCOPY listing information about a BASIC program

PROCMAN

This is another of my efforts, similar in terms of what it sets out to do to QCOPY, but handling standard untokenised programs instead. The name Procman stands for Procedure Manager.

There are two versions, one for pointer driven systems, the other for non-pointered systems.

The pointer version needs the Menu Extension from Jochen Merz, but as that's supplied with most of his QL software, most users are likely to have it already.

Procman asks you to select a BASIC program, then loads it and produces a list of procedures and functions for you to select. Tell it which routines you want to extract and it will try to build an output file consisting of the selected routines. In other words, you can use it to extract and build up a library of routines for your programs. Extract whole routines from your or other people's programs for inclusion in your own (copyright permitting of course).

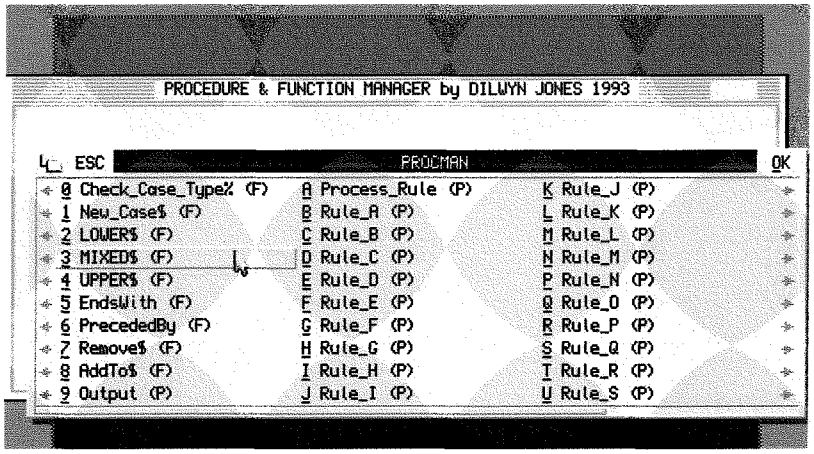


Figure 11 - Selecting procedures/functions in Procman

Once you've selected the routines, you can choose to send the output (the extracted routines) to a file, to the printer to get a printout of the routines, or to the SCRAP system if you wish to paste the routines to other programs. You can also stuff the filename of the output file to the stuffer buffer so that you can just Alt-Space in Basic or another program to load the file generated by Procman.

Event though I wrote this little utility myself, I'm quite proud of it. It could do with being updated to work a little better with modern systems (it can sometimes mess up the display by outlining a 512x256 screen rather than just the size of its own window) and perhaps remove the dependency on the Menu Extension, but it works well enough and I use it a lot - it is very handy for creating libraries of BASIC routines.

Procman can be downloaded from my website at **http://www.dilwyn.me.uk/program/index.html**

Better Basic

One of the many programs released by QL programming legend, Chas Dillon. This was published by Digital Precision as the "Better Basic Expert System". They described it thus: "SuperBASIC is a super BASIC. If you want to improve your programs automatically, and learn as you do this, get Better Basic."

Looking past the hype, this is a program which will take a look at your SuperBASIC program (I am uncertain if it will work correctly with SBASIC, as it was written before the days of SBASIC, although it seems to work) and try to tidy it up.

It asks you where to load the program from (doesn't really understand level 2 directories but can just about cope with them) and save it to. Generally, the source BASIC should have the extension _bas and the saved reworked file will have the extension _rnm. It can list a file as it works, showing what it's up to, can renumber the lines, reformat the listing - indent the lines so you can clearly see where loops and structures start and end and add some warning text to the program if you wish - and limit the maximum length of lines if you wish.

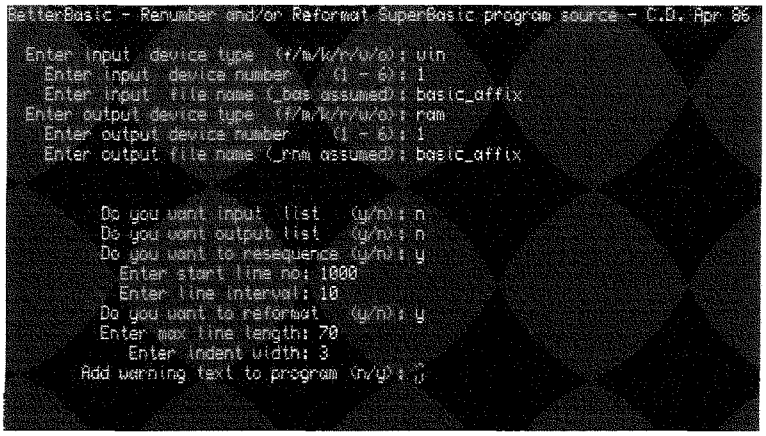


Figure 12 - Better Basic Expert System

While it does a pretty good job as it stands, the source BASIC for this package is now available at **http://www.dilwyn.me.uk/cdillon/index.html** for anyone wishing to update it for more modern systems, or make it rewrite BASIC programs in ways better suited to individual programming styles. A version compiled with a more up to date version of Turbo is available - if you have the Turbo compiler the source BASIC includes all compiler directives to make it compile pretty easily.

LNAMES

One of those horrible tasks facing BASIC programmers who include toolkits in their programs is trying to work out what extension names the toolkit uses, to try to ensure that you don't use clashing names in your program. This is where Adrian Ives's LNAMES program comes in. Basically, all it does is look at a toolkit file (or any file of extensions) and try to list all the names it contains. It will also try to identify which are procedures and which are functions - the latter are flagged by a pair of brackets after the name. Figure 13 shows the main display of the program, while Figure 14 shows a typical output text, in this case after the program tackled my Display_cde toolkit. The list of keywords can be saved as a plain text file, or copied to the Menu Extension's Scrap system. ListNames is a pointer driven program which needs the Menu Extension.

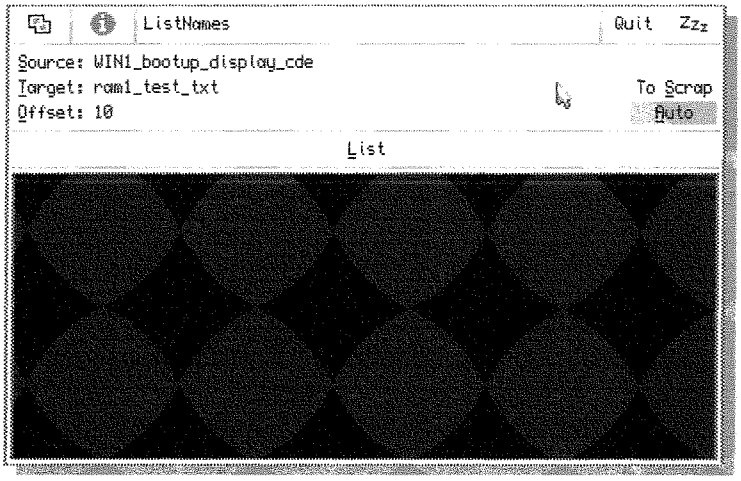


Figure 13 - ListNames program

```
Scan of file: WIN1_bootup_display_cde

Size: 1100 bytes
File version: 2
Last updated: 2009 Nov 08 21:24:11
Offset to Names: 10

$00A0 MOVEMEM
$00E0 ADDRESS()
$0124 BYTES()
$0178 DMODE()
$0198 SYS_VAR()
$01C6 FLIM_X()
$01CE FLIM_Y()
$01B6 FLIM_W()
$01BE FLIM_H()
$022A PTR_ENV()
$026A WIN_MAN()
$033A GD2()
$02AC PTRVER$()
$0314 OS_VER$()
$0390 WMAVER$()
```

Figure 14 - Sample output from ListNames

TRACE programs

There are a number of trace programs that you can get for SuperBASIC. One such program is Strace or SuperTrace from Stack Software. This is now freeware and can be downloaded from the programming software page on my website:
http://www.dilwyn.me.uk/program/index.html

Other such utilities are available to Quanta members through the group's Software Library.
What these do is to monitor which line and statement number SuperBASIC is executing at the time and display the numbers on screen in a small window. Some of these programs will let you set "breakpoints" in the BASIC program, so that you can pause the program at that point and study what's going wrong in that section of the program for example.

A trace utility may be of limited use if all it does is display line and statement numbers. Some programs may be so convoluted that a rapidly changing display of line numbers is not really helpful. Some of these programs will let you pause BASIC, a few might let you "single step" the BASIC program, which makes it easier to follow what's going on. Facilities differ from program to program, so it's hard to give a recommendation as to which is best to use. Some programs include the source code (usually in assembler) so experienced programmers may be able to modify the code slightly to achieve their desired results.

You may have noticed that I have referred to these trace programs in the context of SuperBASIC. Sadly, the BASIC variables used by these trace programs are not used in SBASIC, so these Trace programs are doomed to failure on SMSQ/E systems.

Structured SuperBasic (SSB)

SSB is Tim Swenson's development system for S*BASIC users. Rather than being just a programming aid, it's more of a programming environment. It changes the way you write BASIC programs for the QL.
S*Basic is a structured language. It's designed to be entered with line numbers, but if you use structured programming techniques, without using line numbers and GOTO commands, you can just as easily write S*Basic programs in a text editor, as is done with C and other programming languages.

This is the core function of SSB. You write programs in a text editor, without line numbers, and optionally splitting the program into several separate modules as you might do with a C or assembler program. The SSB program then does the hard work of turning it into a usable BASIC program, or getting it compiled.

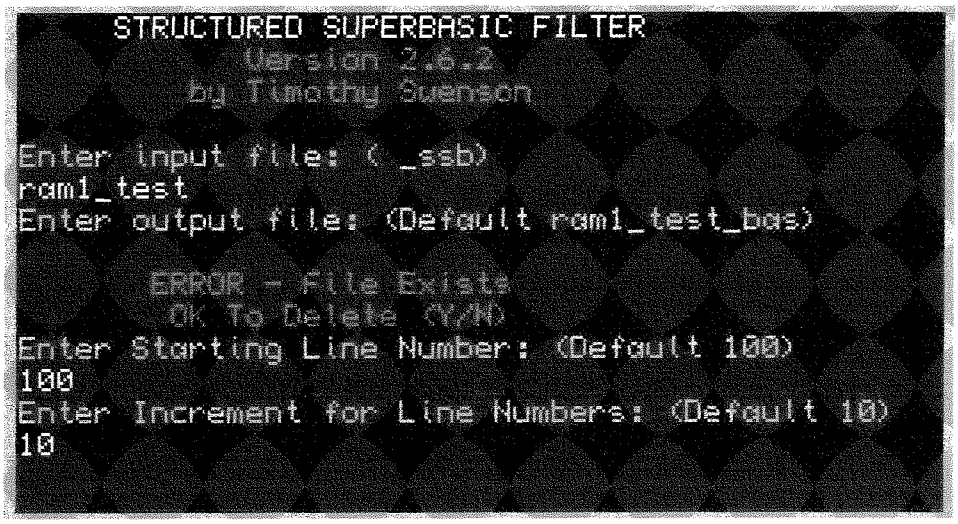


Figure 15- The SSB executable.

It has the advantage that you can create and edit a BASIC program in an editor with the freedoms that entails - not being tied down to line numbers, use of a text editor's features such as search, replace, copy, paste and so on, ability to write programs as modules which can be linked together later and only committing to an actual S*BASIC program once you have written and typed in the necessary parts of the code. There are some additional directives (commands specific to SSB) you can include to control the BASIC code generated - more about this below.

The slight downside of this is that it is a change in the way of working which may take a while to get used to, plus of course a text editor can't check S*BASIC syntax, so you have to wait for that until you get to make it into an actual S*BASIC program later. So while it can be a powerful way to write modular BASIC programs, you do have to get used to a very different way of doing things. It can be quite useful for people like me who tend to have a library of core routines for my programs, which I can just join to my programs with a #INCLUDE directive in the main SSB file I am writing for my program.

A program called ssb262_exe is executed to convert the SSB program text file to a S*BASIC file. This takes the SSB file(s) you created and builds the output BASIC program. It asks you for input file (the SSB file you created), the output file to create, and the first line number and increment steps - usually the program would start at a line like line 100 and every line would have a number higher by 10 than its predecessor.

In addition to allowing programs to be written as multiple modules, SSB allows a degree of conditional compilation using #DEFINE directives in the SSB program file. The author gives the example of a program which needs to use different codes to run on SuperBASIC systems, Minerva MultiBasic systems and as an SBASIC job. Subsequent #IFDEF directives can then cause different code to be included in the output BASIC file generated.

In addition to what I have mentioned so far, there is also a program called SSBGO to automate the use of SSB and the Qliberator BASIC compiler. This runs a program through SSB, loads the program into SuperBASIC, saves the program from SuperBASIC, runs a utility called GREP to scan the resultant program generated for any "MISTake" keywords inserted into the SuperBASIC program by the interpreter (usually means you got the syntax of that line wrong!), and finally starts Qliberator to compile the program. Automation like this helps make the effort of learning the SSB system worthwhile.

In fact the author goes even further and describes using the MAKE utility from C68 with SSB. I won't say much else about this as I am not familiar with MAKE myself.

The author also describes the use of the Revision Control System with SSB, to keep track of changes in software versions as you develop them over the years.

Conclusion

While SuperBASIC and SBASIC are simple to use programming languages which allow us to write our own programs for the QL and its compatibles quickly and easy, I hope I have shown that there are some very useful and simple to use programs and tools out there which can add to our programming experience and generally make life easier and add facilities we may not have considered before.

Gee Graphics - Part 49

by H.L. Schaaf

Last time we mentioned a spigot algorithm* that generated digits of e, and now we will add it to the algorithm for digits of PI.
I've mucked about with the previous listing 'SpigotWander_bas', and if you merge this listing with it you will hopefully have a new and improved(?) program. As before the radix and number of digits wanted are entered. Then the PI spigot is turned on by the PROCedure 'get_PI_digits' and then the e spigot is opened by the PROCedure 'get_e_digits'. The PROCedure 'wander' shows the Brownian(?) paths for the digits of PI, the digits of e, and the combined mix of digits from both PI and e.

* *A.H.J. Sale*
"The Calculation of e to many significant digits"
Computer Journal, Volume 11, 1968, pages 229-230

```
100 REMark Spigot_PI_e_Wander_wip
110 REMark HL Schaaf May 1, 2010
120 REMark for QL Today, GG#49
135 REMark ref: AHJ Sale 1968
200 n99$ = radix -1
210 IF radix›10 : n99$ = CHR$(54 + radix)
230  n99$ = radix - 1
261 PRINT \'getting PI'
262 get_PI_digits
263 PRINT \'getting e'
264 get_e_digits
268 DEFine PROCedure get_PI_digits
380 FOR j = 1 TO (dw + 10)
680 :
800 :
840 :
900     d$ = d$&FILL$(n99$,nines)
920 :
1020 decimalize pi$
1022 END DEFine get_PI_digits
1025 :
1030 :
1060 :
1180 :
1210 :
1211 angle = 36
1212 IF pow‹›INT(pow): angle = 360/base
1213  PRINT #2;dw;" QL random digits
     0-";(360/angle)-1
1215 INK 6
1270  FOR i = 1 TO dw+1
1272   IF pi$(i)='.' : NEXT i
1275   IF CODE(pi$(i))‹58 THEN
1277    TURNTO angle * pi$(i)
1278   ELSE
1280    TURNTO angle*(CODE(pi$(i))-55)
1285   END IF
1290   TURNTO#2, angle*RND(0,(360/angle)
       -1)
1300   MOVE 1
1310   MOVE #2,1
1320  END FOR i
1330 INK 2 :POINT 0,0 : PENDOWN
1335 PRINT dw;' digits of e'
1340 FOR i = 1 TO dw+1
1341  IF e$(i) = '.' : NEXT i
1342  IF CODE(e$(i))‹58 THEN
1343   TURNTO angle * e$(i)
1344  ELSE
1345   TURNTO angle*(CODE(e$(i))-55)
1350  END IF
1355   MOVE 1
1360 :
1365 END FOR i
1368 combine
1372 INK 6
1376 PRINT #0;"radix = ";radix
1400 DEFine PROCedure decimalize (pie$)
1440  dp = '.' INSTR pie$
1450  FOR i = 1 TO LEN(pie$)
1490   IF pie$(i) INSTR('0123456789') THEN
1500    pv = pie$(i)
1520    pv = CODE(pie$(i))-55
1570 IF pie$==pi$ AND sum == PI : EXIT i
1575 IF pie$==e$ AND sum == EXP(1): EXIT i
1585  PRINT "decimal check ";sum
1600 DEFine PROCedure get_e_digits
1610 kf = radix
1620 e$ = '2.'
1630 IF kf = 2 : e$ = '10.'
1640 dw% = dw
1650 start = DATE
1660 pow = LOG10(kf)
1670 base = 10^pow
1680 m% = 4
1690 test = INT(dw%+2)*LN(base)
1700 REMark PRINT 'test = ';test
1710 REPeat loop
1720  m% = m% + 1
1730  chk = m%*(LN(m%)-1)+.5*LN(2*PI*m%)
1740  IF chk › test : EXIT loop
1750 END REPeat loop
1760 m% = m% +2
1770 PRINT 'size of array is ';m%
1780 DIM coef%(m%)
1790 FOR j = 1 TO m% : coef%(j) = 1  : END
     FOR j
1800 FOR i = 1 TO dw%
1810  carry% = 0
1820  FOR j = m% TO 2 STEP -1
1830   temp = coef%(j)*base + carry%
1840   carry% = temp DIV j
1850   coef%(j) = temp MOD j
1860  END FOR j
1870  IF (pow == INT(pow) AND (pow › 1))
      THEN
```

```
1880   FOR j = pow TO 1 STEP -1
1890    IF carry% < 10^(j-1) : e$=e$&'0'
1900   END FOR j
1910  END IF
1920  IF carry% >= 10  AND (pow<>INT(pow))
1930   e$ = e$ & CHR$(55+carry%)
1940  ELSE
1950   carry$ = carry%
1960   IF LEN(carry$)< INT(pow) THEN
1970    carry$ = FILL$('0',(INT(pow)-1)-
        LEN(carry$))& carry$
1980   END IF
1990   e$ = e$ & carry$
2000  END IF
2010  PRINT #0;e$
2020  IF LEN(e$)>(1+dw%) : EXIT i
2030 END FOR i
2040 e$= e$(TO 1+dw%)
2050 PRINT \e$\LEN(e$)-1;' digits in
     ';DATE - start;' seconds'
2060 decimalize e$
2070 END DEFine get_e_digits
2080 :
2090 DEFine PROCedure combine
2100 INK 4 :POINT 0,0 : PENDOWN
2110 PRINT dw;' combined digits of PI & e'
2120 FOR i = 1 TO dw+1
2130  IF (e$(i) = '.') OR (pi$(i) = '.') :
      NEXT i
2140 IF CODE(e$(i))<48 THEN
2150  e_digit = e$(i)
2160 ELSE
2170  e_digit = CODE(e$(i))-55
2180 END IF
2190 IF CODE(pi$(i))<48 THEN
2200  pi_digit = pi$(i)
2210 ELSE
2220  pi_digit = CODE(pi$(i))-55
2230 END IF
2240  TURNTO angle * e_digit
2250   MOVE .5
2260  TURNTO angle * pi_digit
2270   MOVE .5
2280 END FOR i
2290 END DEFine combine
2300 REMark :  End of Listing
```

Floating Point on the QL

by George Gwilt

Computers can easily do calculations on numbers provided they are integers. This usually means that the numbers range from -32768 to +32767. Such numbers fit into one word of 16 bits. Of course the answers to any operation must be within that range otherwise an overflow error will occur. Multiplication, addition and subtraction all obviously lead to integral answers. In the case of division, where the answer could well contain a fraction, the result is presented by the computer as an integer and a remainder. The integer is the rounded integral part of the true answer and the remainder is what's left.

Floating Point Format

If you are using pencil and paper, you can easily write down integers outside the range -32768 to +32767. A hundred digit number just takes a bit longer to write than one of three digits. If you want to write down a number with a fraction, you just put a point between the whole number and the fractional part. But how does a computer cope? The answer is that it uses floating point notation to represent numbers outside the integer range allowed. In this notation you have two elements, an integer, called a mantissa, and an exponent. Thus, the number 2000 in floating point notation could be:

```
2E3
```

The integer, or mantissa, 2 is multiplied by 10 raised to the power 3.

This representation is not unique. The number could equally well be written as:

```
20000E-1
```

Technically speaking, as I have indicated, the integer in these examples is called the mantissa. As you can see, in these examples the same number is represented if the mantissa is multiplied by the kth power of 10 and the exponent is reduced by k.

Since a computer does not have infinite storage, both the mantissa and the exponent must be restricted in size. This implies that a floating point number is not always exactly equal to the number it represents. To make the number as accurate as possible, the mantissa is taken to hold as many significant digits as possible. This holds for all numbers except zero which is represented by a zero exponent and zero mantissa.

In the QL the mantissa is a signed number of length 32 bits. You will see then that bit 31 of the mantissa must not equal bit 32. Apart from that the mantissa can contain any array of bits. the range of values is thus

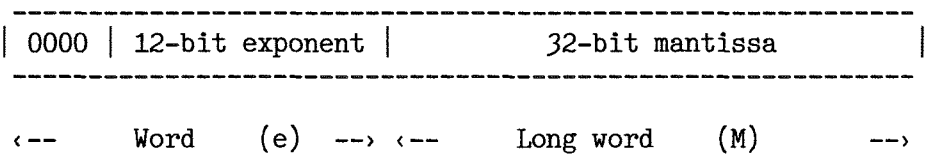
```
2^30 to 2^31-1 and -(2^30+1) to -(2^31)
```

The exponent is a 12-bit number, e, ranging from 0 to 4095. The actual exponent, E, say is calculated as:

```
e - 2079.
```

Thus the actual exponent ranges from -2079 to 2016.

The index and mantissa are held in a three word space as shown below.



So, if the mantissa has value M, the actual number represented is:

```
2^(e - 2079)*M
```

For example the number 1 would be $080140000000.

The calculation is 2^($801-2079)*$40000000
or 2^(2049-2079)*2^30
or 2^-30*2^30
which is 1.

Floating Point Arithmetic

Now that we have a way of representing numbers outside the common integer range it might be interesting to see how to perform the simple operations of addition, subtraction, multiplication and division.
Perhaps the easiest operation is multiplication. It is fairly obvious, I hope, that the way of doing this is to multiply the mantissae (or is it mantissas?) ***[Ed. mantisssae]*** and add the exponents. The result of the multiplication will have to be scaled so that it fits into a 32 bit long word and the exponent will have to be adjusted accordingly.
Division is much the same as multiplication. That is, you divide one mantissa by the other, making sure that you take enough places to allow the answer to be scaled. One exponent is subtracted from the other and then adjusted according to the scaling of the answer's mantissa.
Addition and subtraction are perhaps more tiresome. If the exponents of the two numbers are the same, then the operation takes place on the two mantissae, which is easy enough, though the answer may have to be scaled. If one exponent is smaller than the other, then the number with the smaller exponent must be scaled so that its exponent equals that of the other. Then the operation can take place on the mantissae.
You can see from this that floating point arithmetic is not wholly straightforward.

The QL Assembler Solution

If you are using SuperBASIC or SBASIC (S*BASIC) all the complexities of floating point arithmetic are hidden from sight. For example you can write:

```
z = x + y
```

or

```
z% = x% + y%
```

without even being aware that the first operation requires floating point arithmetic while the latter does not.

You might think that an assembler program using floating point variables would be pretty complex compared with one dealing just with integers. This is not the case with the QL. Since software had to exist to allow floating point arithmetic in S*BASIC it was relatively easy to adjust the operating system to allow assembly programmers access to this. The result is the two vectored routines RI_EXEC and RI_EXECB.
I first used these routines years ago, but stopped doing so when I bought a Thor 21 which had a floating point co-processor, the 68881. Later on some QXLs and later on still the Q40 and Q60 came along with FPUs. So, having acquired these machines, I continued to use the FPU for floating point routines in assembler language.
Recently, however, I have been using QPC2 which is quite a lot faster than even the Q60, but QPC2 does not (yet?) emulate a Motorola FPU so I now have reverted to the RI_EXEC and RI_EXECB vectors for floating point operations.
You may well wonder why anyone would go to the trouble of using assembler for these calculations considering how easy it is to use S*BASIC. The answer is speed. If you want to produce Mandelbrot pictures using your own program you will find that S*BASIC, even with QPC2, is pretty slow. By CALLing an assembler routine you can gain useful speed in an S*BASIC program.
The same applies to several other similar applications.
Of course you can program the whole operation in assembler but this may not be everyone's cup of tea.

What I want to do now is concentrate on RI_EXECB. I hope to explain what it can do, how you would use it and show that it is quite simple to use it for Mandelbrot calculations.

RI__EXECB

The workings of both RI_EXEC and RI_EXECB are explained by Norman Dunbar in Part 16 of his series on Assembler. This appears in QL Today Vol 11 Issue 2 Sept-Nov 2006.

Briefly then, here is what the registers must contain on entry to the vector:

```
A1,A6.L   points to the maths stack
A4,A6.L   points to the base of the variables area
A3        points to the operation codes
```

The program to set this going is:

```
MOVEA.W   RI_EXECB,A2
JSR       (A2)
```

(I know that it is suggested that D7.L should be made zero, but that is only for some pre-JS roms and even then only for some operations.)

Maths Stack

The floating point numbers on which the operations take place are on the maths stack. The first item on the stack is called TOS (Top Of Stack), and the next number is called NOS (Next On Stack). There may well be other numbers on the stack but they are not acronymically defined.
In S*BASIC the pointer to the maths stack (relative to A6) is held in BV_RIP(A6)

Variables Area

In S*BASIC it can be useful to use the Basic Buffer as the variables area. The address of this buffer (relative to A6) is at the address found in A6.
The variables area is intended for the storage of floating point numbers which can be transferred to and from the maths stack.

The Operation Codes

The codes available in SMSQ/E are a combination of the original ones in the QL and the new ones invented for Minerva. These are listed in various places and not all of them are correct. I think that the ones listed here are correct for SMSQ/E. The codes with odd values are those from Minerva and must not be used in QL operating systems.

Value	Mnemonic	Description
$00	qa_end	End of list
$01	qa_one	Push 1 (float)
$02	qa_nint	Round fp to nearest word integer
$03	qa_zero	Push 0 (float)
$04	qa_int	Truncate fp to integer
$05	qa_n	Push following byte (-128 to 127)
$06	qa_nlint	Round fp to nearest long integer
$07	qa_k	Push constant according to following byte
		Following constants are:
$56	qak_pi180	pi/180
$69	qak_loge	log10(e)
$79	qak_pi6	pi/6
$88	qak_ln2	ln(2)
$98	qak_sqrt3	sqrt(3)
$A8	qak_pi	pi
$A7	qak_pi2	pi/2
$08	qa_float	Integer word to fp
$09	qa_flong	Long word to fp
$0A	qa_add	Add TOS to NOS
$0C	qa_sub	Subtract TOS from NOS
$0D	qa_halve	TOS/2
$0E	qa_mul	TOS*NOS
$0F	qa_doubl	TOS*2
$10	qa_div	NOS/TOS
$11	qa_recip	1/TOS
$12	qa_abs	Absolute value
$13	qa_roll	(TOS)B,C,A -› (TOS)A,B,C roll 3rd to top
$14	qa_neg	Negate
$15	qa_over	NOS-›TOS (A,B -› B,A,B)
$16	qa_dup	Duplicate
$17	qa_swap	NOS‹-›TOS
$18	qa_cos	cosine
$1A	qa_sin	sine
$1C	qa_tan	tangent
$1E	qa_cot	cotangent
'20	qa_asin	arcsine
$22	qa_acos	arccosine
$23	qa_atan2	arctangent(NOS/TOS)
$24	qa_atan	arctangent
$26	qa_acot	arccotangent
$28	qa_sqrt	Square root
$29	qa_squar	TOS*TOS
$2A	qa_log	Natural log
$2C	qa_l10	Base 10 log
$2E	qa_exp	Exponential
$30	qa_pwrf	NOS^TOS
$32	qa_pi	Push pi

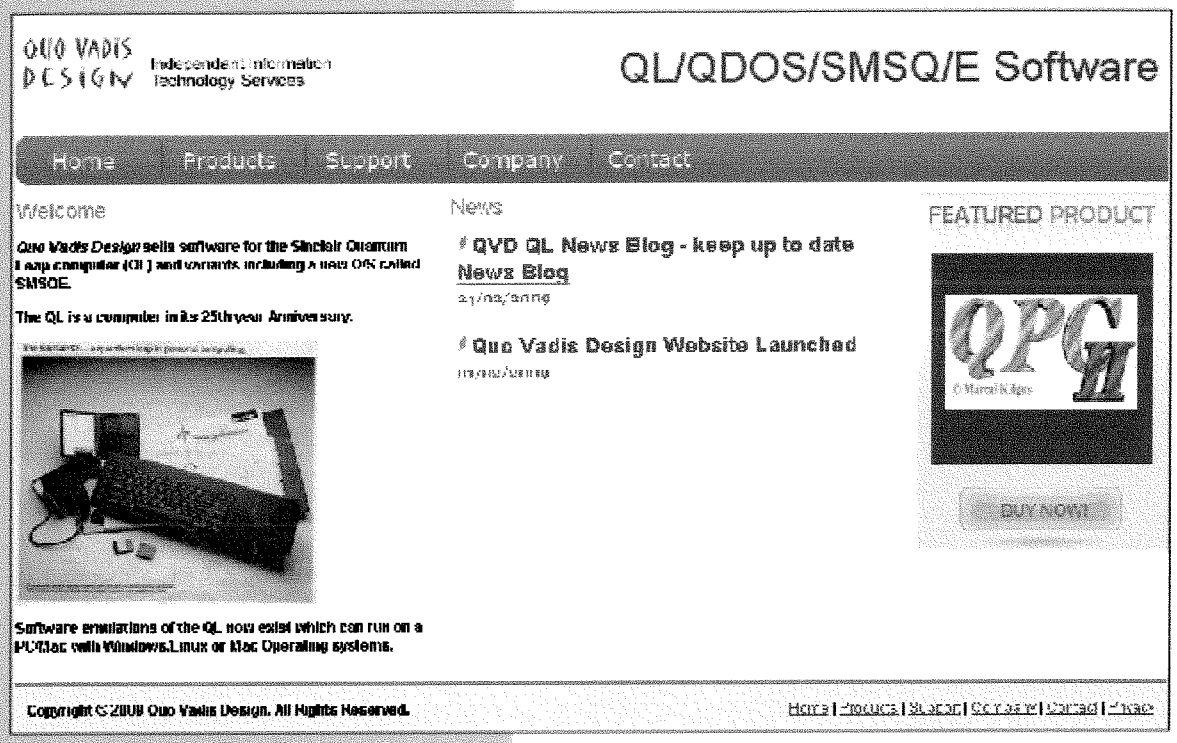
Codes $34 to $FF are used to transfer floating point numbers between the variables area and the maths stack.

The even codes are augmented to negative words by the addition of $FF00 giving a range of $FF34 to $FFFE or -204 to -2. These are used as an offset to (A4,A6.L) to give the address of the floating point number in the variables area which is transferred to the top of the maths stack. If one is added to these codes the result is that the number is transferred from the top of the maths stack to the variables area. Thus the code $FA will result in the number at -6(A4,A6.L) being put on the stack. $FB will result in TOS being placed in -6(A4,A6.L).

QUO VADIS DESIGN

Independant Information Technology Services

www.ql-qvd.com



Check the QL News Blog on our website for updates.

www.ql-qvd.com/blog

Bruce@ql-qvd.com

Quo Vadis Design
38 Derham Gardens
Upminster
RM14 3HA
UK

Tel: +44 (0)20 71930539
Fax: +44 (0)870 0568755

You will see from this that there is more than one way of accessing a particular address in the variables area. If a particular address has offset F from A6 and if K is the value of A4 then the offset from (A4,A6.L) is R where

```
R + K = F
```

There are, in general, several pairs of R and K whose sum equals F. This implies that there is not a unique way of defining and using the variables area.
The method I use is this. For example, to produce a space for N numbers I will need 6*N bytes. I set (A4,A6.L) pointing to the byte just beyond the end of the 6*N bytes. The offsets from (A4,A6.L) for each of the numbers is thus -6, -12, -18 and so on. The codes to put these numbers on the maths stack are thus $FA, $F4, $EE and so on.

For clarity I use special codes for transferring numbers between the maths stack and the variables area. Thus for a variable called var1, say, I define g_var1 as the code to transfer var1 to the stack and p_var1 as the code to transfer var1 from the stack. So, if I have three variables var1, var2 and var3 in the variables area the equates to define g_var1 etc are:

```
g_var3   equ      -6
g_var2   equ      g_var3-6
g_var1   equ      g_var2-6
p_var1   equ      g_var1+1
p_var2   equ      g_var2+1
p_var3   equ      g_var3+1
```

This assumes that the numbers are set in the variables area in the order var1, var2 and var3, with A4,A6.L pointing to the byte just beyond the end of var3.

Mandelbrot Example

To show how one string of codes can perform several operations I describe the iteration needed for one pixel of a Mandelbrot diagram.

The basic iteration is;

```
z' = z^2 + a
```

where z and a are complex numbers. The iteration is repeated a pre-determined number of times or until z escapes from a circle of radius 2.

If we write:

```
        z = p +iq
and     a = x + iy
```

the iterations are:

```
        p' - p^2 - q^2 + x
and     q' - 2*p*q + y
```

I assume that the variables stored are:

p	the current value of p
q	the current value of q
p2	the current value of q^2
q2	the current value of q^2
x	
y	

Here are the operations needed:

```
Op        TOS                 NOS
g_p2      p^2                 -
g_q2      q^2                 p^2
qa_sub    p^2 - q^2           -
g_x       x                   p^2 - q^2
qa_add    p^2 + q^2 +x        -                   NOTE: TOS = p'
qa_dup    p'                  p'
qa_squar  p'^2                p'
qa_dup    p'^2                p'^2
p_p2      p'^2                p'                  NOTE: p2' stored
g_p       p                   p'^2
g_q       p                   q
qa_mul    p*q                 p'^2
qa_doubl  2*p*q               p'^2
g_y       y                   2*p*q
qa_add    2*p*q + y           p'^2                NOTE: TOS = q'
qa_dup    q'                  q'
p_q       q'                  p'^2                NOTE: q' stored
qa_squar  q'^2                p'^2
qa_dup    q'^2                q'^2
p_q2      q'^2                p'^2                NOTE: q2' stored
qa_add    p'^2 + q'^2         p'
qa_n, 4   4                   p'^2 + q'^2
qa_sub    p'^2 + q'^2 - 4     p'
qa_swap   p'                  p'^2 + q'^2 - 4
p_p       p'^2 + q'^2 - 4     -                   NOTE: p' stored
0         p'^2 + q'^2 - 4     -                   NOTE: Finished
```

This leaves the amount to be tested as the only item on the stack.

I should perhaps make it clear that some of the operations used here are available with SMSQ/E but not with QDOS. The results can be obtained in a different way using QDOS. For example the results of qa_swap can be replaced by defining two variables temp1 and temp2, say and using:

```
p_temp1, p_temp2, g_temp1, g_temp2
```

Defining the Variables Area

A - With Executable Programs

A good place for the variables area in an executable program is its dataspace, especially if A6 is made to point to its start. This can be done by

```
        lea     (a6,a4.l),a6
```

being issued near the program's start.

We can reserve space for four variables, for example, at the start of the dataspace by setting

```
vars    equ     4*6
```

The instruction

```
        lea     vars,a4
```

will set the pointer to the variables area.

B - With Code to be CALLed it is probably best to use the Basic Buffer.

As already been mentioned the address of the Basic Buffer, relative to A6, is found at (A6). Thus, to set A4 to the base of the variables area we have:

```
        movea.l (a6),a4             Address of buffer relative to A6
        lea     3*4(a4),a4          A4 points to the end of the area
        cmpa.l  8(a6),a4            Compare the buffer end with A4
        bhi     oops        --->
```

The last two instructions test that the Basic Buffer has enough space.

25 Years - Part 3

by Tony Tebby

Welcome to the next part of our series. This time, we look at the time before Windows appears ... and this is where we continue in the next volume ... enjoy!

25 Years - 1988 to 1990 - Sci-Fi, Worms and Unix

Worms in the system

In 1988 viruses had been circulating freely attached to programs on PC diskettes for several years. For even longer, the academic and science fiction worlds had been toying with ideas for "free living" agents or worms that travelled around networks of computers carrying out maintenance tasks.

1988 - The Morris Worm

The Morris worm turned sci-fi into reality. This worm brought down many of the VAXs and Sun 3s running BSD Unix connected to the Arpanet. Different people have interpreted the incident in different ways, but for me the incident brought the degenerate state of the IT "establishment" to my attention. I am not referring to Robert Tappan Morris, the author of the worm, but the computer centre managers and system administrators of sites with Unix machines connected to the Arpanet.

The worm was analysed in a report[9] from Purdue University:

> Although UNIX has long been known to have some security weaknesses (*[citations going back to 1979]*), the scope of the break-ins came as a great surprise to almost everyone ... The most noticeable effect was that systems became more and more loaded with running processes as they became repeatedly infected. As time went on, some of these machines became so loaded that they were unable to continue any processing *[about 20 processes]*.

Why should this have been a great surprise to anyone? The risks had been known for about a decade. Even earlier, in 1975, Dennis Ritchie had written "The first fact to face is that UNIX was not developed with security, in any realistic sense, in mind"[10]. The machines affected were mainly in academic and research establishments and in the US Department of Defence and they were very lucky as the worm did not carry a malicious payload. Morris created and released the worm, but it was the computer centre managers and system administrators that created the conditions that allowed it to spread. It appears that, although Morris was convicted, no computer centre manager or system administrator lost his job or was convicted of criminal negligence for connecting Unix systems directly or indirectly to a public network. This failure to take action against those who were really responsible for the damage caused by the attack was to have serious consequences.

1990 - Desert Storm

Two years after the Morris Worm, the US launched Operation Desert Storm and Operation Desert Shield. The ground had been well prepared: in the intervening period, a group of hackers had placed backdoors in a number of military, DoE and DoD servers, using known Unix security flaws, giving them almost complete freedom to access the US military networks. This time the "worms" had payloads, not to destroy the systems, but to download everything about US military plans and capabilities. It appears that the operation was run for profit from the Netherlands, but it was not profitable as they could not find any buyers for the information. Had all the potential buyers been there first?

When I first heard about this story, I had my doubts, surely those responsible for US military networks could not be so stupid or negligent that they continued to use Unix machines? But then I came across the US Navy Computer Incident Response Guidebook[11]

9 http://homes.cerias.purdue.edu/~spaf/tech-reps/823.pdf
10 http://www.tom-yam.or.jp/2238/ref/secur.pdf
11 http://all.net/books/ir/nswc/P5239-19.html

> Many of the cases of unauthorized access to U.S. military systems during Operation Desert Storm and Operation Desert Shield were the manifestation of espionage activity against the U.S. Government.

Apparently, the "espionage" attacks were not the majority of "unauthorized accesses" and that does not include the intrusions that were not detected - it is just mind boggling.

Everyday worms and other attacks

Worm attacks continued, carrying payloads such as trojan horses, key loggers and denial of service co-ordinators. The situation took a turn for the worse when the world's largest software supplier abandoned its own system software in favour of a remodelled Unix system. With the arrival of Windows NT, Microsoft's systems became vulnerable to the same type of attacks as mainstream Unix, culminating with the "Code Red" worm in July 2001 which cost computer users an estimated $3B.

The Morris and Code Red worms exploited fundamental design weakness in the Unix / C execution model to break programs executing on the target computer, but other attacks exploited human fallibility.

In 2001 / 2002, Gary McKinnon "scanned a large number of computers in the .mil network, was able to access the computers and obtained administrative privileges ... McKinnon would then use the hacked computer to find additional military and NASA victims" (US Department of Justice). With administrative privileges, he could have done enormous damage, but he was only "looking for evidence of UFOs". How did he get in? Just by using remote login with blank or obvious passwords!

In 2003, H. Orman wrote, in The Morris Worm: A Fifteen-Year Perspective[12],

> Today, the Morris worm is remembered as the first of many such attacks, as what might have been a wake-up call to system administrators and security researchers, and as the first certain signal to those who still held utopian beliefs about the Internet, that it was not to be a friendly place.

"Might have been a wake up call", but it was not. System administrators just dozed on with "business as usual".

> As the years passed, knowledge about subverting Unix access permissions abounded and spread. The number of loopholes, and their varieties, had begun to look unmanageable to many system administrators and computer-security experts. Two camps developed, one hoping to fix all the problems, and another advocating keeping one step away from the Internet.

Why only two camps? Surely the obvious solution was to ditch Unix. A number of attempts were made to develop "replacements", but, in general these were Unix rebranded, Unix restructured or Unix rewritten. Even though there were numerous private attempts to create "better" systems, there were no significant attempts to deploy systems that were radically more efficient, radically cleaner, radically more predictable or radically more secure.

1990 - Plan 9

Plan 9 (the title of a sci-fi film) was the Bell Labs Unix team's attempt at getting it right the second time around. What is most striking was the difference between quick and dirty design of the original Unix and the X Window System, and the "we are going to get it right" philosophy of the Plan 9 development. The story in Box 8 is as it is told by the developers[13]. Remember, when you read this text, it was written by the creators and developers of Unix: there is, therefore, a certain bias! I did not write Unix, therefore there is a certain bias in my comments.

Apart from the "Unix centric" approach, this discussion starts off very reasonably, explaining why the central time sharing concept of the 1960s had lost favour and why the personal computer systems that were replacing it were, themselves, far from satisfactory for large organisations. Then it slips away and they end up aiming to build straightforward, archaic 1960s time sharing systems with central machines and terminals! Or rather they aimed to emulate this archaic systems architecture using cheap or not so cheap microcomputers.

12 IEEE Security & Privacy September/October 2003
13 http://plan9.bell-labs.com/sys/doc/9.html

As the discussion goes on it becomes clear how little functional difference there was between Unix and Plan 9: the same kernel concept, the user based access rights, the same hierarchical file system *and the same vulnerabilities.*

Scattered throughout the text are the keywords "complete" and "consistent". Compared to the earlier Unix and X Window System philosophies of not caring about either, this is a welcome return to rigorous design. The authors noted, with some surprise, that taking a general, consistent design approach actually saved time when it came to extending the capability. Why should they have been surprised? Because by the time they came to extend Plan 9, the belief that rigorous, complete and consistent design can save you time had come to be considered as a sign of incurable mental illness or senility (see Worse is Better).

Some quotes (circa 1991) from Ken Thompson and Rob Pike of Unix and Plan 9 Fame.

Object oriented programming

- Object-oriented design is the Roman numerals of computing
- We have persistent objects: they're called files

Structured programming

- If you want to go somewhere, goto is the best way to get there

Unix

- Not only is Unix dead, it's starting to smell really bad.

X

- The X server has to be the biggest program I've ever seen that doesn't do anything for you.

With attitude like that, they cannot be all bad.

Box 8 - Plan 9 rationale and implementation

Plan 9 began in the late 1980's as an attempt to have it both ways: to build a system that was centrally administered and cost-effective using cheap modern microcomputers as its computing elements. The idea was to build a time-sharing system out of workstations, but in a novel way. Different computers would handle different tasks: small, cheap machines in people's offices would serve as terminals providing access to large, central, shared resources such as computing servers and file servers. For the central machines, the coming wave of shared-memory multiprocessors seemed obvious candidates *[see "shared memory multiprocessing" below]*. The early catch phrase was to build a Unix out of a lot of little systems, not a system out of a lot of little Unixes.

By the mid 1980's, the trend in computing was away from large centralized time-shared computers towards networks of smaller, personal machines, typically Unix 'workstations' *[How isolated from reality can you get? Unix had negligible penetration outside the academic world]*. People had grown weary of overloaded, bureaucratic timesharing machines and were eager to move to small, self-maintained systems, even if that meant a net loss in computing power *[Most people, as they were not using Unix, saw a massive net gain in computing power]*. As microcomputers became faster, even that loss was recovered, and this style of computing remains popular today.

In the rush to personal workstations, though, some of their weaknesses were overlooked. First, the operating system they run, Unix, is itself an old timesharing system and has had trouble adapting to ideas born after it. Graphics and networking were added to Unix well into its lifetime and remain poorly integrated and difficult to administer *[the comment is also true of MSDOS, the dominant workstation OS at the time]*. More important, the early focus on having private machines made it difficult for networks of machines to serve as seamlessly as the old monolithic timesharing systems. Timesharing centralized the management and amortization of costs and resources; personal computing fractured, democratized, and ultimately amplified administrative problems *[this fails to draw the very important distinction between centralised data and centralised processing - and their relative merits in different types of organisation - another example of one-size-fits-all]*.

...

The problems with Unix were too deep to fix, but some of its ideas could be brought along. The best was its use of the file system to coordinate naming of and access to resources, even those, such as devices, not traditionally treated as files. *[Ouch!]* ... our laboratory has a history of building experimental peripheral boards. To make it easy to write device drivers, we want a system that is available in source form *[what system, other than Unix, required you to have the OS source in order to write device drivers?]* ... [Instead of] normal Unix-style processes and light-weight kernel threads, Plan 9 provides a single class of process but allows fine control of the sharing of a process's resources such as memory and file descriptors *[Just like Stella and not far from Domesdos c.f. Box 2]*. A single class of process is a feasible approach in Plan 9 because the kernel has an efficient system call interface and cheap process creation and scheduling *[but still orders of magnitude slower than Stella]*.

25 Years - 1991 to 1994 - Better or Worse, Windows, Linux

25 Years - 1991 - The Rise of "Worse is Better"

"The rise of worse is better" is a chapter of Richard Gabriel's lament for Lisp[14] published in 1991. This chapter, which circulated as a article in its own right, was clearly intended to be provocative and he adopts the devil's advocate method of argument. It has often been quoted as the definitive explanation for the dominance of very, very, bad system software from the mid 1980s onwards.

He attempted to explain why, although Unix and C are clearly "worse" than "his" AI operating system and Lisp compiler, the Unix+C pair was apparently more successful. He compared the "New Jersey" (Bell Labs) "Worse is Better" philosophy with the MIT "Right Thing". These two philosophies are detailed in Box 9.

The New Jersey approach is described as providing a quick and dirty, incomplete solution which is then evolved by armies of developers working independently: ultimately the best solution is that which attracts the most developers. MIT "Right Thing" approach is described as leading to the interminable development of slow, overly complex, oversized, inefficient programs.

There are some serious flaws in the arguments. The MIT / New Jersey distinction is nonsense. The X Window System was quite deliberately designed breaking every one of his "Right Thing" rules - even more so than Unix - and X was created at MIT - whereas Plan 9 was designed using a strict "Right Thing" philosophy - and Plan 9 was from New Jersey.

The comparison between C and Lisp is no better. Two years before the article was written, C had become mature enough to be standardised as ANSI X3.159-1989, "only" 20 years after its creation, whereas Lisp, created 10 years before C, was still evolving and would not be standardised until three years later as ANSI X3.226-1994. Lisp is, therefore, a better example of the "New Jersey" approach than C.

Box 9 - MIT versus New Jersey by Richard Gabriel

The essence of the [MIT/Stanford] style can be captured by the phrase **the right thing** ... it is important to get all of the following characteristics right:

- Simplicity – the design must be simple, both in implementation and interface. It is more important for the interface to be simple than the implementation.
- Correctness – the design must be correct in all observable aspects. Incorrectness is simply not allowed.
- Consistency – the design must not be inconsistent. A design is allowed to be slightly less simple and less complete to avoid inconsistency. Consistency is as important as correctness.
- Completeness – the design must cover as many important situations as is practical. All reasonably expected cases must be covered. Simplicity is not allowed to overly reduce completeness.

I believe most people would agree that these are good characteristics. I will call the use of this philosophy of design the MIT approach

The [Unix] worse-is-better philosophy is only slightly different:

- Simplicity – the design must be simple, both in implementation and interface. It is more important for the implementation to be simple than the interface.
- Correctness – the design must be correct in all observable aspects. It is slightly better to be simple than correct.
- Consistency – the design must not be overly inconsistent. Consistency can be sacrificed for simplicity in some cases, but it is better to drop those parts of the design that deal with less common circumstances than to introduce either implementational complexity or inconsistency.
- Completeness – the design must cover as many important situations as is practical. All reasonably expected cases should be covered. Completeness can be sacrificed in favor of any other quality. In fact, completeness must sacrificed whenever implementation simplicity is jeopardized. Consistency can be sacrificed to achieve completeness if simplicity is retained; especially worthless is consistency of interface.

14 http://www.dreamsongs.com/NewFiles/LispGoodNewsBadNews.pdf

His description of the two basic *right thing* development scenarios is even more peculiar.

> The big complex system scenario goes like this: First, the Right Thing needs to be designed. Then its implementation needs to be designed. Finally it is implemented. Because it is the Right Thing, it has nearly 100% of desired functionality, and implementation simplicity was never a concern so it takes a long time to implement. It is large and complex. It requires complex tools to use properly. The last 20% takes 80% of the effort, and so the right thing takes a long time to get out, and it only runs satisfactorily on the most sophisticated hardware.
>
> The diamond-like jewel scenario goes like this:
>
> The right thing takes forever to design, but it is quite small at every point along the way. To implement it to run fast is either impossible or beyond the capabilities of most implementors."

In a rebuttal[15] Richard Gabriel did point out the he was not supporting "Worse is Better" and he came close to apologising for calling Unix a virus, but his premise remained intact: trying to design software is totally futile - whatever you do it will turn out oversize and underperforming and the more complete, coherent and correct you try to make it, the longer it will take to release.

He does not appear to consider the classic "Right Way" scenario.

> First, the Right Thing needs to be designed. Then its implementation needs to be designed. Finally it is implemented. Because the scope is known from the start it has 100% of the design functionality. Because no short cuts are taken on implementation, no reworking is required, saving time. It is compact, clearly structured and efficient. The regularity of the interface makes it easy to use. Because most of the time is spent building solid foundations, the last 80% is a downhill run so it finished quickly and the system needs only the minimum of resources to run.

It is clear from the Plan 9 article cited above, that the authors of Plan 9 believed that they had implemented The Right Thing in the Right Way. Having been involved in dozens of development projects (for quite a few, called in at the last minute panic stage), I have to admit that projects using a Right Thing design philosophy may not always follow the Right Way scenario, but they will never result in a big complex system. The most likely problem is that doubts set in half way through the project when there is still (quite rightly) no visible sign of life. At this point there is a risk the the project veers off into the quick hack ("Worse is Better") approach, the software starts ballooning, the errors start multiplying and the completion date starts receding. What is worse, this is justified as "pragmatism" to get the system out "even if it is not perfect".

Richard Gabriel also illustrates these two development philosophies on a specific example: the "PC loser-ing" problem. The MIT approach to solving the problem was to add complex and fragile code, after the problem had been identified, to give the "right" results. The Unix "Worse is Better" approach just gets it wrong but flags the error so that applications can take the necessary corrective action (or else crash).

The MIT solution was clearly not the "Right Thing" because if the system had been designed according to the Right Thing philosophy (the correctness rule), the problem would not have occurred: Domesdos was designed to be "correct" so "PC loser-ing" was impossible. This took some thought in design but, because there was no "tricky" code or extra error handling, it was faster to implement, it was more compact, it was faster in execution and the applications did not need to allow for an error that could not occur: in this case, it was the "Right Thing" done the "Right Way".

Finally, he concludes that the Worse is Better approach wins over the Right Thing because the Right Thing delays the release and winner is the first to be released, no matter how incomplete, incoherent or difficult to use it may be. Even on this he is clearly wrong, Lisp was widely used a decade before anyone outside Bell Labs had ever heard of C, but Lisp lost out to C.

He seems to have missed out on the point: that, in the beginning, Unix and C were freeware. Even if they were worth less than you paid for them, most organisations' accounting methods did not, and still do not, consider time wasted as an expense. In academic establishments, time wasted is "research". Lisp and Lisp machines cost money - they never stood a chance.

15 http://dreamsongs.com/Files/worse-is-worse.pdf

Behind the rhetoric, there is, however, something far more sinister: in computer science courses, around the world, students were being taught that writing correct, complete, coherent software was either impossible or not worth doing and that users are unimportant.

A year before Richard Gabriel's rant, Professor A Tannenbaum published his "Modern Operating Systems" in which he wrote on the evolution of operating systems "this evolution is very similar to the evolution from assembly language programming, where the machine came first, to programming in high level languages, where the programmer came first – programmers are no longer able to write compact, efficient software[16]".

This is an extraordinary "insider view" of computer science. In the "bad old days" when programmers were able to write compact, efficient *and reliable* software, they did not write this compact, efficient *and reliable* software for the machine itself, but wrote compact, efficient *and reliable* software *for the users of the machine*.

1991 - The Rise and Fall of Virtual Memory

Although Unix workstations had suffered from virtual memory for many years, it was only introduced to mass market personal computers in Mac System 7 in 1991 and Windows 3.1 in 1992

There are three popular views of virtual memory and at least one controversial view.

1. Virtual memory allows you to execute programs that need more than the physical memory of the computer.
2. Virtual memory increases hardware costs and reduces performance.
3. Virtual memory allows the performance to degrade gracefully as demand for memory outstrips supply.
4. Virtual memory increases the demand for real memory.

The idea of virtual memory being virtually unlimited is a simplistic view believed by too many software developers, but it is vaguely true, provided you do not care at all about usability.

That virtual memory is costly is inescapably true. Virtual memory requires additional hardware over and above that required for memory access protection and all accesses to data and programs are slowed down - even under the most favourable conditions.

The concept of degrading gracefully depends on your definition of "graceful". 25 years ago this was a idealistic view of virtual memory, but is now completely unrealistic.

That virtual memory increases the demand for real memory is totally contrary to the theory, but observably true. There must, therefore, be something wrong with the theory.

The failings are discussed in Box 10.

The end of virtual memory in sight

In 2009, there seems to be a certain amount of dispute amongst "tweekies" as to whether it is a good idea turning off virtual memory on systems such as Windows.

Some tweekies recommend it, others trot out conventional wisdoms such as "you gain no performance improvement by turning off the pagefile", clearly without trying it. My own experience on a rather memory limited portable is that overall performance is significantly better as I no longer suffer from slow motion windows. Most software that I use runs perfectly. The only significant problem is that, from time to time, Firefox gives up with out of memory, usually when it is just sitting around doing nothing. This is not serious, as it can be restarted, re-opening all the tabs while occupying a few hundred megabytes less memory.

With diskless computing becoming more common, we can possibly look forward to a future without virtual memory. In the shorter term, double your RAM every one or two years to keep ahead of virtual memory guzzlers.

16 This is my own translation of the French translation of a book written in English by a Dutchman.

Box 10 - The failings of virtual memory

Disgraceful degradation

Virtual memory does allow programs executing to take more memory space than is really available. The principle is that not all the memory space taken is actually used, having the unused or little used space "swapped out" to disk releases valuable "real memory" for use. The memory is divided into pages. If a page is in memory, it can be accessed almost as fast as if real memory were used. If there is a "page fault" because the processor tries to access a page that has been swapped out, there is a long delay. The delay does not depend on the memory overload, but the page fault rate does.

25 years ago, a typical average disk access time was about 40ms with memory cycle times of about 400ns: a ratio of about 100,000 times. A one in ten thousand page fault rate would have caused a computer to run 10 times more slowly than the "benchmark" speed: this could be considered graceful degradation.

In 2009, an optimistic typical average disk access time is about 6ms with L1 cache access times of 1ns: a ratio of about 6,000,000 times. A one in ten thousand page fault rate will cause a modern desktop computer to run 600 times more slowly than the "benchmark" (L1 cache) speed: this cannot be considered to be graceful degradation.

What is the memory overload that will cause a one in ten thousand page fault? There is no simple answer as it depends on the details of the memory usage of all the programs executing. Experience with a fairly large variety of PCs from Windows 95 onwards (business and private usage) has indicated that users tend to think that their machine is suffering from viruses (slow, erratic execution) when the memory overload reaches 20%-30%.

For at least a decade, the standard cure for PC performance problems has not been to increase the processor speed, but to increase the size of the RAM to avoid using virtual memory.

The costs of virtual memory

Virtual memory requires a dynamic address translation unit (more commonly and incorrectly known these days as a memory management unit, MMU, although it does not do any memory management). This translates the "virtual addresses" seen by the processor into the real memory addresses and raises a page fault error if the virtual address is not in real memory.

25 years ago, MMUs were more expensive than processors. In 2009, they are built into the processor bus interface and, possibly, account for less than 20% of the processor cost.

The address translation in the MMU is not free, it takes time to calculate the real memory address (if there is one) for each and every access to data or program. 25 years ago, this typically added 20% to the memory access time under the most favourable conditions. In 2009, it not only adds overheads to external memory accesses but complicates caching as well.

Virtual memory increases the demand for real memory

"We all" know that this is true, but why?

The glib answer is in Parkinson's law as applied to computer memory: "programs expand to fill the available memory". The result is that the virtual memory space fills up, overloading the real memory. Is this inevitable? Why should this happen? Why should this be serious?

On introducing virtual memory, there is an immediate effect and a more pernicious longer term effect.

In the the short term, memory overload occurs with applications written BVM (before virtual memory) for two main reasons.

1. Well behaved software, such as the older versions of Microsoft Word, will check whether there is enough room before taking extra temporary working memory. Without virtual memory, fall back methods are used if there is not enough real memory. With virtual memory, extra memory always seems to be available so it will be taken and the availability of virtual memory will automatically increase the demand for real memory.
2. With virtual memory, users can start applications for which there is not enough room in real memory without stopping other applications. This results in instant memory overload which can make it almost impossible to remove any of the applications as the machine has become so slow. (In one extreme case I had to deal with a PC that that had been "corrupted" by a minor software upgrade: it took more than 25 minutes to start up - the user thought it was dead, but it was just a simple memory overload).

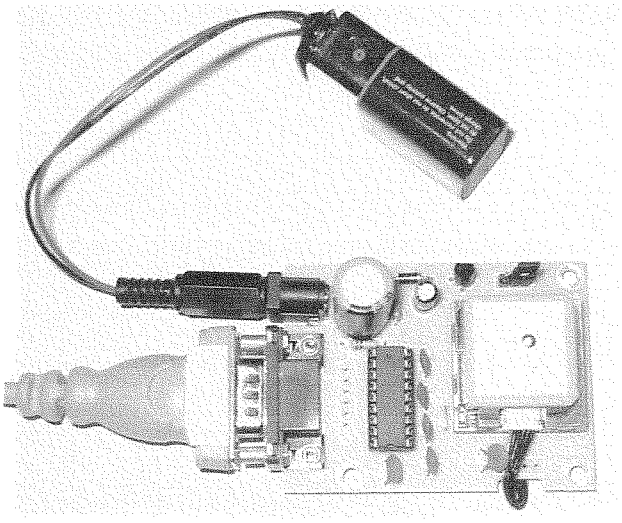
In the longer term, memory overload occurs with applications written AVM (after virtual memory) because many developers have adopted the attitude that memory is unlimited and they just do not care.

Improving QL Emulator I/O, Oscilloscope

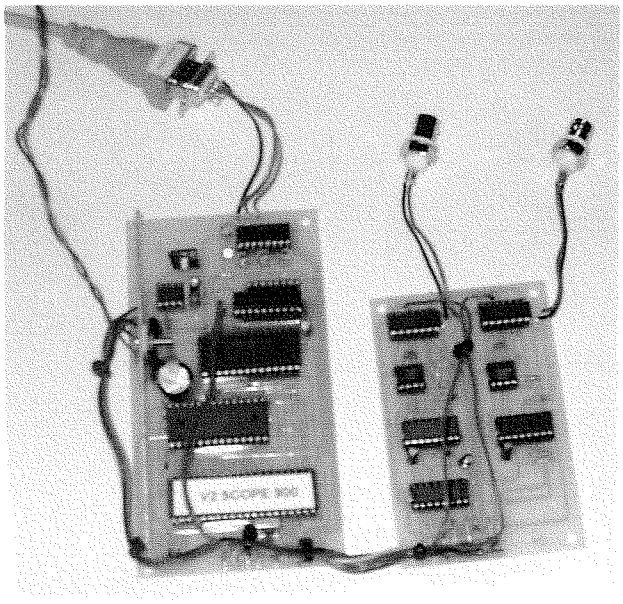
by Ian Burkinshaw

[Editor:] First, two apologies to Ian: one for having held back his second article for two issues - but there's a good reason for this: we do not have very many hardware articles, so we thought it would be a good idea to spread it a bit.
When I started with the layout of this article, I noticed that I used the wrong picture to go with Ians GPS article in Issue 1 of Volume 14.
The correct picture, which should have been printed together with the article of issue 1, is shown to the right.

And now enjoy the next of Ians very interesting hardware articles...



In my first article on improving the QL emulator I/O published in volume 12, issue 3 of QL Today I showed how you can get parallel inputs and outputs via the RS232 port on your PC. As promised in that article here is the second part dealing with the oscilloscope project. This, in the event took me longer than I expected, sorting out routines to carry out the FFT (Fast Fourier Transform). More on this later.
The hardware for this project was originally published in the August and September 2007 issues of Everyday Practical Electronics magazine. The project has two PCB's one handles the analogue processing and contains the analogue to digital converters. There are two channels on this card so you have a two channel oscilloscope as a result. It is not impossible to change the entire project to deal with more channels. However you would need to have some PIC programming experience to achieve this. 8 channels should be possible. The basic hardware, with additional input cards can support this. The second card contains the memory and a PIC processor to take the streams of data produced by the analogue to digital converters in to a RS232 stream. There is a lot more data here to be handled, compared to the parallel I/O project. Because of the higher data rates involved, 57,600 baud is required. A PC fitted with a RS232 port or if your PC does not have one then you can use a USB to RS232 converter. However please note, most QL original hardware will not run at these speeds. The analogue to digital converters are able to handle signals up to 40KHz. Not very high by today's standards but for simple audio type projects it is good enough. The signal processing is 8 bit so is not CD quality. Also it is more a storage oscilloscope. I will explain. There is also a mode you can put the hardware into which will make the oscilloscope a 8 bit digital analyser, this aspect I will not be covering in this article, but is not difficult to achieve. As I have said before we are supposed to be tinkerers. It is not my aim to give you a total solution but give you ideas.



The right picture for this article

A storage oscilloscope does what is say on the tin, it stores a snap shot in time of a waveform for you to study in you own time. This also means it is not a real time device. In the case of modern digital oscilloscopes this is very much the case. It takes time for the signal processing to take place, and the computer, in this case a QL emulator, to display the results. Two points to make here, one is, it is not a real time device, that is what you see now did not happen now. Also what you get is also not continuous, it is a snap shot in time, over a fixed time span. So you cannot use this as a continuous waveform streaming device. This is a limitation of this hardware solution. Even higher data rates would be required to achieve this and I do not think even a QL emulator could keep up with this. Do not under estimate how much processing power is required for even relatively low frequencies that audio has (20KHz). Taking into account you need to sample a signal at twice the original signal frequency. That is why an audio CD runs at 44.1KHz, it is just over what is required to give you a 20KHz bandwidth, which is the limit of human hearing. The main problem is you have to process on a continuous basis. Remember we have to acquire the data, process it and display (output) it. It all takes time.

If you can solder then you can make this project. The PCB and the components are easily available. The PCB's from Everyday Practical Electronics(1) publishers and most of the components from Farnell(2) for example, but there are others places such as RS, Rapid etc. If you do not have the ability to program a PIC (Microcontroller chip), then pre programed PIC's can be obtained from Magenta Electronics(3), who advertise in the magazine. You can use the same power supply that you may have used for the serial controller card.

The instructions in the original EPE article are fairly good, I built mine from the instructions to make sure they were accurate for this article, and did not find any problems and my boards worked first time. However I must come clean and say I am an electronics engineer by training so I do have an advantage, but I tried to look at this as if an inexperienced constructor. They even give you resistor colour codes to follow. Do watch for the position and orientation of components. However there is one error. The part list: Note IC15 should be a 79L05 (not 78L05), the circuit is correct.

Also some individual set-up of PC's can prevent the V2 PC Scope from working with its serial control. This can be cured by installing the entire EPE Serial software from the download area of the EPE web site at;

ftp://ftp.epemag.wimborne.co.uk/pub/PICS/SerialOCX/

Accessible via the Downloads section of their home page **www.epemag.co.uk**

Download all the files into a temporary folder, make sure you have no other applications running (including virus checkers, email clients, Visual Basic ect and run the SETUP.exe and follow the prompts. This will install a copy of the OCX; and all its sub-components and correctly install and register them. The OCX software has been tested with Windows 98, 2000, ME and XP. You may only need to do this with the original EPE Visual Basic software.

The listing is not a final program but it does give you a basis to work from. It will display both channels and show you a spectrum display from channel one. At start up you can select simulate, enter "S" at the prompt, so you do not need the hardware connected to give you an idea of what can be achieved. If you do have the hardware the enter 'R' for Real, then you will be asked which serial port you are using. If you look at the my program listing it will give you a good idea how to control the card. But here is some additional information to help you.

Send "G" 'GO' tell the PIC to get ready to send
Receive "R" Wait for an 'R' to be sent from PIC (Indicates Ready)
Send "B" Tell the PIC to send a block (256 samples, 8 block per channel, 16 in total)
Send "S" Tell the PIC to get ready to receive setting data (Gain, Bias and AC/DC coupling)
Receive "V" Valib Baud, could not get this to work on my own code or the original Visual Basic program. But is not important unless you wish to change the baud rate. Not a good idea anyway.

No LF and/or CR on "G", "R", "B" or "S" commands.
Data returned does have LF and CR terminators.

You need to send some other data after the "S" start command. In the following order. AC/DC coupling & X10 attenuator this is a single number see below. The second character to be sent is for Gain 'Channel A' followed by Bias 'Channel A', Gain 'Channel B', Bias 'Channel B',Mode (See below) and then "G" to end the string.
The data for the gain and bias controls are from 0 to 255.

The bias setting is an offset value so for example it can deal with positive and negative input values, so for example to display an AC signal such as a sine wave it would be best to set the bias to 127 so the analogue to digital converter works at it's mid range, so the AC single will swing either side of this value.
Gain settings can be anything you like from 0 to 255. But as a guide the following values will give you some known gain settings.

23=/10
48=/5
84=/2
127=X1
170=X2
213=X5
233=X10

The first piece of data after the "S" start command set the AC/DC coupling and the attenuator. This is all controlled as a 8 bit word in to a 8 bit output port in the hardware as follows:

1=Channel A X10 attenuator (Bit 1)
2=Channel A AC/DC coupling (Bit 2)
4=Channel B AC/CD coupling (Bit 3)
8=Channel B X10 attenuator (Bit 4)

Just add together the numbers you require, so for example send '12', will set the AC/DC coupling to DC and the X10 attenuator to 'On' again for channel B.

The mode command near the end of the string is as follows:
0=Oscilloscope Mode
1=Spectrum Mode
2=Digital Analyser Mode
Note: In this listing the spectrum mode is not used, but does need to be sent.

Another aspect I have included in the listing is FFT (Fast Fourier Transform) so that you can have spectrum displays. This is more a library of routines. I claim no originality for these they are transcribed from the routines in the "The Scientist and Engineers guide"(4). This also goes into details of how Fast Fourier Transforms work so I will not repeat here. It is not perfect but does give a good idea what can be done in terms of signal processing and display on the QL platform.

Sources and References

I have no connection with any of the organisations below. They are just what I have used in the preparation of this article.

(1) "Everyday Practical Electronics" magazine
http://www.epemag.co.uk
(2) "Farnell" source of electronic components
http://www.farnell.com
(3) "Magenta Electronics" source of pre-programmed PIC's
http://www.megenta2000.co.uk
(4) "The Scientist and Engineer's Guide to Digital Signal Processing", Chapter 12.
http://www.dspguide.com/pdfbook.htm

The listing is rather long and would fill about 8 or 9 pages of this magazine. As promised at the start of this volume not to print long listings anymore but make them available at
www.QLToday.com
I have forwarded the listing to Bruce Nicholls who maintains this website and I am sure you will be able to download it from there by the time you read these lines.

Letter-Box

George Gwilt writes:
Letter to QL Today re Norman Dunbar's Article Easy PEasy Part 1

Although Norman claims that most of the article is based on what I wrote in a readme file I have two small comments.

1. On page 15 the contents of D1 are shown as the position of the pointer. This is quite correct, but it is essential to know that the co-ordinates are absolute and not relative to the program's window or any of its sub-windows.
2. At the top of page 16 the table indicates that D4.B contains the "Event number" and that D2.W contains the "Event code". This is quite correct but the text above wrongly shows the reverse. The event numbers look peculiarly arbitrary. The numbers 16 to 23 are in fact the bit numbers in the event vector corresponding to the events. This event vector is the long word at $14 of the window status area, whose address is at wst0.

Artificial Intelligence

by Stephen Poole

In America, there are two views of how the Universe originated. The oldest is the biblical description in the book of genesis, called 'creationism'. The second is the scientific description of evolution, which applies not only to living things, but also to all forces and matter.

Creationists consider that life is so complex that it must have been created perfectly by an intelligent God, that is, using 'intelligent design'. Science has analysed most aspects of reality in far greater detail, and has come to the conclusion that the universe is so old that the mere forces of nature combined with probability have had the time to generate ever more complex structures into what we see today. Take the case of man: For creationists man is perfect, being in "God's image", but for Science man is imperfect, as continuous and numerous birth defects prove, but on average, evolution has given man enough variability to survive from one generation to the next, imperfect as he may be.

In a military sense, intelligence is just masses of raw data collected far and wide. For the ancient Greek, Asiatic and Jewish philosophers, intelligence was the quest for Wisdom, distilled from generations of study and debate of all gathered knowledge, honed into a set of recommendations of how to live life in a harmonious way. Religious intelligence is none other than a vast knowledge of all the world's sacred texts.

Modern psychologists test people's brains to evaluate their IQs, (intelligence quotas), which allows them to measure their ability to come to logical deductions, which is probably most peoples' idea of what intelligence means. IQ is not a measure of Wisdom or general knowledge, just a person's potential to become intelligent as he grows older.

We could study the 'clever' way stars produce all known elements from hydrogen, but this would be to adopt creationists notions of intelligent design, (which leaves room for little comment). Instead, we can look at the more productive attitude of Science, which examines which individual natural laws canalise matter and energy within set semi-random bounds to produce organised organic matter: in particular the DNA helix, the matrix of life. Recently, genetics has proved the veracity of evolutionary theory beyond all doubt, by comparing the genes of many species.

DNA can be thought of as intelligent, as it sorts through the genetic code in a cell, choosing the best genes and repairing damaged ones. Viruses are the simplest life-forms, but as they are parasitic we shall not consider them here, as they are incapable of independant existence. So let's move on to bacteria:

These life-forms, simple as they are, reveal an astonishing adaptation to most environmental conditions, from freezing temperatures to boiling ones, from light to totally dark situations, from acid to alkaline waters, and from rich organic food to metallic chemical 'food', and from lightweight to intense pressure situations. From a Darwinian standpoint, adaptability is the main feature of intelligence, but can we say that bacteria think? If we observe the way in which they move towards food sources, avoid dangers and use various collective strategies to survive we might well assume that they are indeed astute.

On the next evolutionary level we may consider polycellular animals such as sponges or plants. These respond very actively to different essential stimuli and can communicate with one another using external chemical signals to indicate food sources or predators, much in the same way that internal human organs communicate using hormones. They have no nerves so no brains, but they do have collective organisation to a great extent.

One step further up the evolutionary ladder lie vertebrates with a spinal chord, nervous system and a rudimentary brain, such as fish. Anyone who has kept fish in an aquarium can testify to the fact that goldfish can be trained to respond to signals from humans and that they have a considerable range of complex behaviour patterns. Unfortunately, we generally think of fish as dull creatures that get easily hooked by lures or as huge shoals that cannot avoid being trapped in nets. (In general we have a pretty poor impression of the animals we eat). The fact is that in nature there are no nets nor hooks, so fish have no fear of them. But in situations familiar to them they can reveal a great deal of cunning and behavioural adaptation.

Then on to birds. Many parrots have been taught to 'speak', as everybody knows, 'Parrot-Fashion'. But this is because man has only recently learned how to teach birds to answer

logically. Crows are excellent problem-solvers that can use all manner of tools to obtain food. They inspect a problem such as a chinese puzzle containing a peanut, turn their heads as they ponder, then work out a strategy to get at the food by solving the puzzle! And all this without language..? Yet recently people have taught African Grey Parrots to say several hundred words, to combine them using grammar and reply logically after reflection, and even to do simple arithmetic... This came as a great shock to many people, as language is supposed to be the thing which seperates man from animals, yet here was proof that animals with minute brains were capable of thought and reflection and even lie-telling!

So on and up another step to Washoe, the bonobo ape who could use a specially devised computer keyboard to interact with her keepers. She even taught her offspring to do the same applying many hundreds of words! Now apes, like wolves are capable of great feats of collective tactics and strategy to obtain their goals, and live in complex societies, like early mankind.

Man was hardly different to other apes throughout great swathes of geological time until he perfected language enabling him to pass on detailed information from person to person in the same, if more complex way that bees or ants do. This enabled him to break the rigid social hierarchy imposed by the dominant males and allowed him to begin to form complex social groups who could make better use of all available resources and construct an important cultural tradition based on exchange. This was vital in hunting and food-gathering groups who were inter-dependant for tool-making and sharing resources such as flints or ochre trading for decoration.

When agriculture began to evolve from simple forest gardening to clearing, planting and harvesting fields of cereals, man began to store food, resources and knowledge enabling massive population growth. Civilisation and cities occurred, and remained largely based on the same tribal rules until the industrial revolution discovered how to harness fossil energy resources, when the world's population exploded liberating numerous scientists to develop knowledge and technology to its current exponential levels. (The downside of intelligence has been global warfare. Even warfare demands intelligence in the form of alliances, but real intelligence stops at such cooperation, thereby avoiding war altogether).

But Fossil fuels and many other essential resources will definitely run out in the near future, creating massive pollution, economic and climate change on a scale unknown to man previously. We can use computer models to predict future trends, but what we need most urgently is the intelligence to face up to the facts, persuade everyone of the need to behave as responsible eco-citizens and make the right choices so as to prove that man is an intelligent species after all.

What does all this have to do with the QL and computing? Well, for the past fourty years there has been sporadic talk of Artificial Intelligence, that is, teaching computers and robots to become the super-brains that will save us all from all of our problems. But how far has this got?

'Big Blue' managed to beat Kasparov in a chess tournament by a short head, but big blue used neither tactics nor strategy to win, just a huge database of all the winning moves during the last thirty years of the world chess series. Machine translation programs are still very poor when compared to human translators as they have no background knowledge of context nor subtelty. Robots are only just beginning to aquire any real degree of autonomy, still requiring some remote-control, and compared to a new-born gnu they are hopelessly handicapped. Self-learning AI systems are mainly good at extracting trade secrets from unsuspecting experts in their respective domains to declassify them in their hierarchy.

So we can see that Artificial Intelligence has a long way to go before it can ape evolutionary intelligence, so we must still rely on good, sound human education to produce results and to ensure the future of our world's civilisation. In my article 'Machine Logic' I mentioned that the human brain contains some 50,000,000,000 neurones. One can add that the human brain in actual fact contains millions of billions of synapse junctions, meaning we are an incredibly long way from matching its performance. In the meantime, we can still study AI to help us, for example to improve internet user-friendliness, which could help old people to keep in touch with their relatives and improve their lives. The $75 One Laptop per Child is now with us and will be on sale in the spring of 2009. And 860,000,000 starving people in this market-based world of ours could certainly benefit from a free internet laptop computer showing them how to escape from their chains of poverty.

Artificial Intelligence programs are designed to make computers think, that is, to make decisions by themselves based on data fed into them. One of the oldest examples of reasoning is the Syllo-

gism of the ancient Greeks. Given two or more statements, one can deduce logical conclusions if both ideas contain common elements and therefore it is possible to make a coherent third statement. This is what this Think_bas program does.

To keep it simple, the routine only acts on a few statements. But see how by searching associations between sentences it can make a whole list of deductions. But its weakness is that it cannot act on hunches like the human brain, so poor old Plato and Diogenes are left out in the cold, as the QL doesn't guess that wise-men are men too...

Feel free to modify the code to add and test extra data such as 'A MAN IS A LAVATORY CLEANER' to see what the QL thinks about Socrates!. (Was Socrates class-concious?). This will show you that 'artificial' Syllogisms only work correctly when you select your vocabulary very carefully.

I was careful to choose statements that are compatible, using but one verbal form. 'A man is a mortal' is much easier to deal with than 'All men are mortals', which, if treated in the same way as with this program, would give the reversed deduction that 'All men are Socrates', clearly absurd. I also used DATA statements to avoid having to parse long sentences. This greatly simplifies the coding, as does replacing spaces with underscores making it clearer for you to understand, but beware of case errors! The crux of the routine is the cross-reference 'flagging' to indicate associations between phrases.

To really impress your friends, instead of just printing the deductions as we do here, get them to type in a question, for example, 'IS SOCRATES A PHILOSOPHER?'. The program should parse the question, inverse the subject and verb, look up the list of QL deductions in the array and prints its reply to the screen. Yes! the beast really is thinking, making deductions and storing its conclusions in memory just as we do....even though in a far less sophisticated way.

The human brain works mainly by the association of ideas, but the main weakness of AI systems is the same as that for translations: Computers do not yet have an adequate mastery of logic and grammar to be able to parse fully efficiently. There is plenty of scope for research in this domain.

Professional expert systems act on hundreds or thousands of statements, and they may also have to use multiple conditions before coming to correct decisions. Therefore they can come to conclusions rapidly, but can still only make the choices for which they are rigorously programmed. So their degree of intelligence is still far below that of the average man....who, (like our womenfolk), are capable of intuition.

It can be argued that the exact definition of intelligence must be taken from Darwin's theory of evolution: It will only be possible to say that man is intelligent after he has survived for hundreds of generations after the present potential environmental crisis. If we extrapolate this definition to computers, computers will be deemed intelligent when they are capable of self-sustained survival. But can you imagine a world where no electronic machinery can become obsolete? We would live in homes jammed full of out of date gadgets that refuse to die. The QL what!

Will Artificial Intelligence really help us to feed the World and avoid an ecological catastrophy? It could scarcely do worse than we are doing now...

```
100 ::
110 REMark illusion_bas, by S.Poole. v22nov08
120  REMark for QL Today. Beta-test by B. Coativy
130 :
140 CLEAR: RESTORE : OPEN#1,con_16
145  WINDOW 512,256,0,0: BORDER: CLS
150  WINDOW 256,206,256,0: PAPER 2
160  m1=1.85: m2=m1*2: m3=m2/8: m11=m1-.5
170  SCALE 80,-8,-16: st=8: ss=2+7*st
180  s=ss-2: r=ss+2: t=90
190 :
200 DATA 0,0,0,0,0,0,0
210 DATA 0,0,3,5,1,0,0
220 DATA 0,3,3,5,1,1,0
230 DATA 3,3,3,5,1,1,1
240 DATA 1,1,1,2,3,3,3
250 DATA 0,1,1,2,3,3,0
260 DATA 0,0,1,2,3,0,0
270 DATA 0,0,0,0,0,0,0
280 :
290 DATA 0,0,0,3,1,0,0,0
300 DATA 0,0,3,3,1,1,0,0
```

```
310 DATA 0,3,3,3,1,1,1,0
320 DATA 0,6,6,6,4,4,4,0
330 DATA 0,1,1,1,3,3,3,0
340 DATA 0,0,1,1,3,3,0,0
350 DATA 0,0,0,1,3,0,0,0
360 :
370 DATA 0,0,0,0,0,0,0,0
380 DATA 0,0,3,3,1,1,0,0
390 DATA 0,3,3,3,1,1,1,0
400 DATA 0,3,3,3,1,1,1,0
410 DATA 0,1,1,1,3,3,3,0
420 DATA 0,1,1,1,3,3,3,0
430 DATA 0,0,1,1,3,3,0,0
440 DATA 0,0,0,0,0,0,0,0
450 :
460 DATA 0,0,0,5,0,0,0
470 DATA 0,3,3,5,1,1,0
480 DATA 0,3,3,5,1,1,0
490 DATA 6,6,6,0,4,4,4
500 DATA 0,1,1,2,3,3,0
510 DATA 0,1,1,2,3,3,0
520 DATA 0,0,1,2,3,0,0
530 DATA 0,0,0,0,0,0,0
540 :
550 FOR up=2 TO ss STEP st
560     FOR ac=6 TO ss STEP st
570         READ sk: square ac,up,7
580 END FOR ac: END FOR up
590 :
600 FOR up=6 TO ss STEP st
610     FOR ac=2 TO ss STEP st
620     READ sk: square ac,up,7
630 END FOR ac: END FOR up
640 :
650 FOR up=2 TO ss STEP st
660     FOR ac=2 TO ss STEP st
670         READ sk: square ac,up,0
680 END FOR ac: END FOR up
690 :
700 FOR up=6 TO ss STEP st
710     FOR ac=6 TO ss STEP st
720         READ sk: square ac,up,0
730 END FOR ac: END FOR up: i$=INKEY$(#1,-1)
740 ::
750 DEFine PROCedure square(x,y,i)
760  LINE x,y: TURNTO 0: MOVE m1: INK i
770  PENDOWN: FILL 1: TURN t: MOVE m1
780  FOR j=1 TO 3: TURN t: MOVE m2
790  TURN t: MOVE m1: FILL 0: PENUP: sq sk
800 END DEFine
810 :
820 DEFine PROCedure sq(qr)
830  IF i=7: INK 0: ELSE INK 7
840  SELect ON qr
850     =1: dot 0  : dot 180
860     =2: dot 180: dot 270
870     =3: dot 90 : dot 270
880     =4: dot 90 : dot 180
890     =5: dot 90 : dot 0
900     =6: dot 0  : dot 270
910     =REMAINDER
920  END SELect
930 END DEFine
940 :
950 DEFine PROCedure dot(dt)
960  LINE x,y: TURNTO dt: MOVE m11: TURN t: MOVE m11: squ
970 END DEFine
980 :
990 DEFine PROCedure squ
1000  PENDOWN: FILL 1: TURN t: MOVE m3
1010  FOR j=1 TO 3: TURN t: MOVE m3
1020  TURN t: MOVE m3: FILL 0: PENUP
1030 END DEFine
```

This is the End ... of this issue AND this Volume!

QL Today's next issue will be ready for you in September but to keep the price low, we need your renewal

NOW!

Thanks for your support!

Time to Renew

Same price as last year!

Customer-Nr. ☐☐☐☐

Name: _____

Street: _____

Town: _____

City: _____

Country: _____

EMail: _____

Subscriptions are also available online from Quo Vadis Design in the UK
www.ql-qvd.com

Full subscription details are also available from
www.qltoday.com

I hereby subscribe to **QL*Today*** for 4 issues of Volume 15. The total price for all four issues is as follows, including postage and packing (depending on destination)

Destination	price for Volume 15
Germany, Netherlands & United Kingdom	EUR 29.90
Rest of Europe	EUR 31.90
Rest of World (airmail)	EUR 42.90

☐ Please charge my credit card: ☐VISA ☐MasterCard ☐Diners Club

☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ Expires ☐☐ - ☐☐

Card Verification Code: ☐☐☐

☐ Money transfer to one of the following accounts:

☐ Deutschland: Jochen Merz, Account 493 50 431, Postbank Essen, BLZ 360 100 43

☐ Österreich: Jochen Merz, Account 85055317, PSK Wien, BLZ 60000

☐ Switzerland: Jochen Merz, Account 60-690080-4, PostFinance, Clearing-Nr. 09000

☐ The Netherlands: Jochen Merz, Gironummer 3258439, Postbank/ING NL Amsterdam

☐ and from all other countries in EUR with IBAN and BIC to account
Jochen Merz, Deutsche Postbank AG, IBAN: DE21 3601 0043 0611 1004 37 / BIC: PBNKDEFF 360

☐ UK customers can pay £26.90 (valid until July 2010) to
Jochen Merz, Account 83795395, Citibank UK, Sort code 30-00-45
or send cheques in £ - no fee for UK sterling cheques (payable to *Jochen Merz* only)!

☐ Paypal: Log into your paypal account and send the money in EUR to paypal@J-M-S.com

Date, Signature _____

Please fill in and send to Jochen Merz Software, Kaiser-Wilh.-Str. 302, 47169 Duisburg, Germany.
or Fax to +49 203 501517 or scan & Email to SMSQ@J-M-S.com

<u>To German-speaking readers with automatic renewal only:</u>
<u>An alle deutschsprachigen Leser mit Auto-Verlängerung & Abbuchung:</u>
Natürlich findet die Verlängerung wie immer automatisch statt!

Time to Renew

Same price as last year!

Customer-Nr. ☐☐☐☐

Name: _____

Street: _____

Town: _____

City: _____

Country: _____

EMail: _____

Subscriptions are also available online from Quo Vadis Design in the UK
www.ql-qvd.com

Full subscription details are also available from
www.qltoday.com

I hereby subscribe to **QL*Today*** for 4 issues of Volume 15. The total price for all four issues is as follows, including postage and packing (depending on destination)

Destination	price for Volume 15
Germany, Netherlands & United Kingdom	EUR 29.90
Rest of Europe	EUR 31.90
Rest of World (airmail)	EUR 42.90

☐ Please charge my credit card: ☐VISA ☐MasterCard ☐Diners Club

☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ Expires ☐☐ - ☐☐

Card Verification Code: ☐☐☐

☐ Money transfer to one of the following accounts:

☐ Deutschland: Jochen Merz, Account 493 50 431, Postbank Essen, BLZ 360 100 43

☐ Österreich: Jochen Merz, Account 85055317, PSK Wien, BLZ 60000

☐ Switzerland: Jochen Merz, Account 60-690080-4, PostFinance, Clearing-Nr. 09000

☐ The Netherlands: Jochen Merz, Gironummer 3258439, Postbank/ING NL Amsterdam

☐ and from all other countries in EUR with IBAN and BIC to account
Jochen Merz, Deutsche Postbank AG, IBAN: DE21 3601 0043 0611 1004 37 / BIC: PBNKDEFF 360

☐ UK customers can pay £26.90 (valid until July 2010) to
Jochen Merz, Account 83795395, Citibank UK, Sort code 30-00-45
or send cheques in £ - no fee for UK sterling cheques (payable to *Jochen Merz* only)!

☐ Paypal: Log into your paypal account and send the money in EUR to paypal@J-M-S.com

Date, Signature _____

Please fill in and send to Jochen Merz Software, Kaiser-Wilh.-Str. 302, 47169 Duisburg, Germany.
or Fax to +49 203 501517 or scan & Email to SMSQ@J-M-S.com

To German-speaking readers with automatic renewal only:
An alle deutschsprachigen Leser mit Auto-Verlängerung & Abbuchung:
Natürlich findet die Verlängerung wie immer automatisch statt!